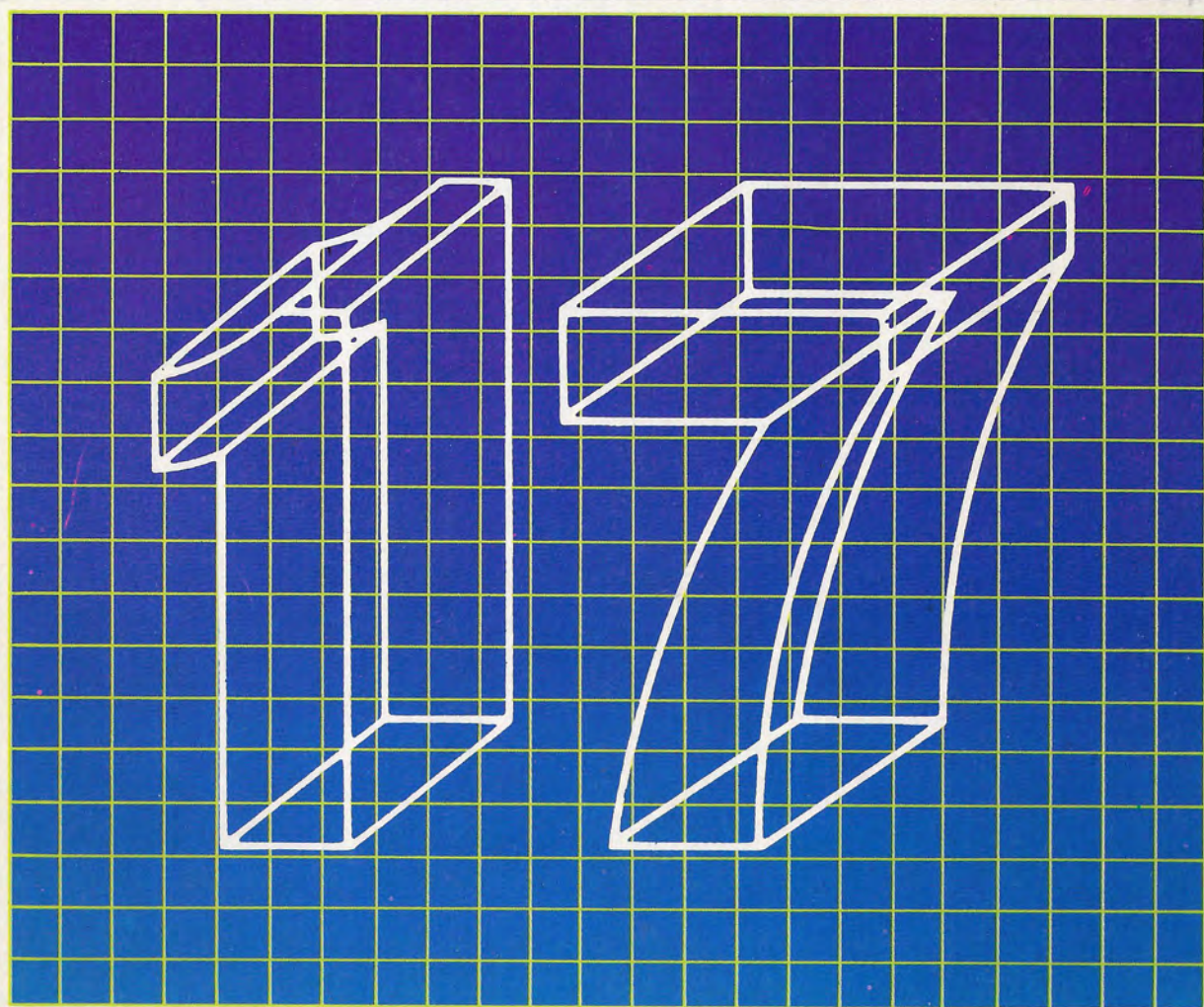


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



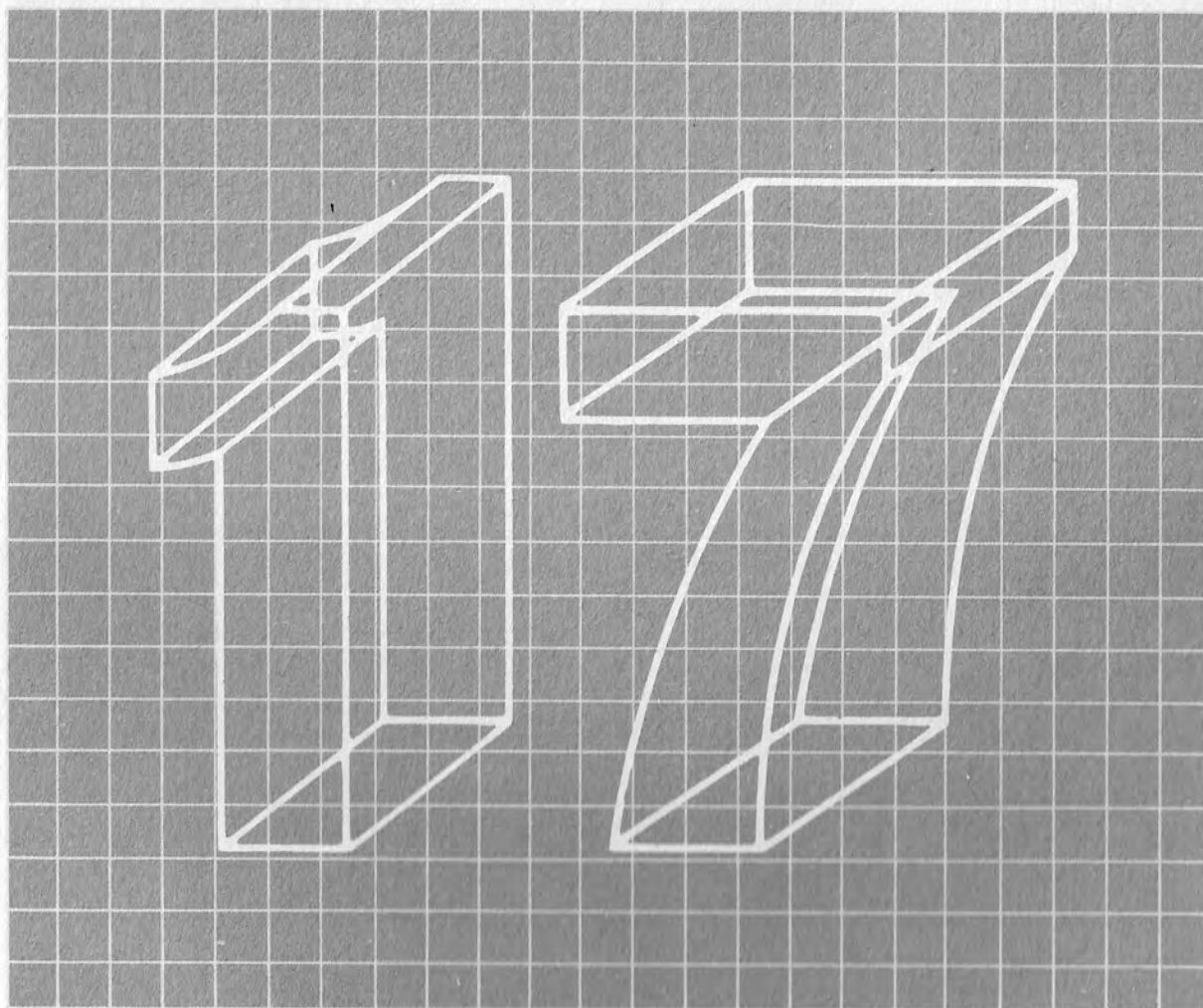
PROGRAMAS EDUCATIVOS,
DE GRAFICOS Y DE UTILIDAD

EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 17. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-109-6

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Mayo, 1987

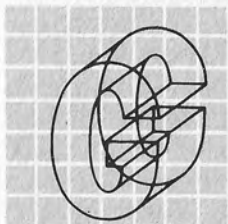
P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	17
■ TRUCOS Y RUTINAS BASICAS	25
■ EL TALLER DEL HARDWARE	31
■ APRENDER CON EL ORDENADOR	39
■ PEQUEÑA HISTORIA DE LA INFORMATICA	49
■ TEMAS MONOGRAFICOS DE VANGUARDIA	52
■ TERMINOLOGIA	56
■ VOCABULARIO DE INFORMATICA	58

EXPERIENCIAS Y PRACTICAS EN LOGO

■ Condicionales



CUANDO queremos o necesitamos que una determinada acción se efectúe o no dependiendo de una determinada condición, utilizamos la orden SI.

Supongamos el siguiente caso:

Una mañana nos levantamos, nos asomamos a la ventana y observamos que está lloviendo. Ante esto, cogemos un paraguas antes de ir a trabajar. Si por el contrario observamos que no llueve no cogemos el paraguas y nos vamos a trabajar.

Veamos cuáles son las diferentes partes de que consta este proceso:

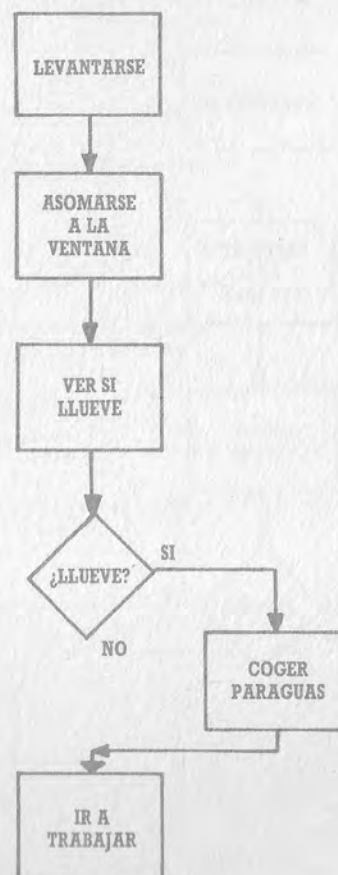
- 1.º Levantarse.
- 2.º Asomarse a la ventana.
- 3.º Ver si llueve.
- 4.º Si llueve coger un paraguas.
Si no llueve, no cogemos el paraguas.
- 5.º Ir a trabajar.

Observa que en el punto cuatro tenemos que realizar una determinada acción sólo en el caso de que esté lloviendo. En este ejemplo, podemos ver cuál es la condición y cuál la acción.

CONDICION: Si llueve.

ACCION: Coger un paraguas.

Representemos gráficamente el caso expuesto por medio de un organigrama:



Esta gráfica expresa claramente cómo, dentro de un proceso compuesto de varias fases, puede llegar un momento en el que aparezca una condición y tengamos que realizar

Si necesitas una función trigonométrica y tu versión de LOGO no dispone de ella, define un procedimiento para obtenerla.

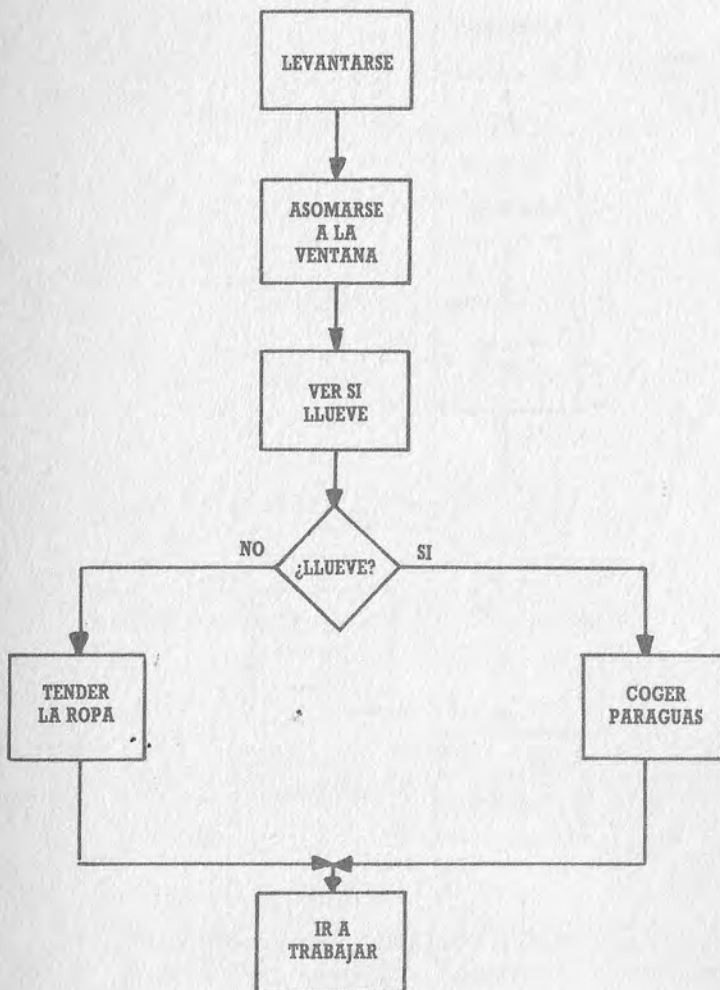
EXPERIENCIAS Y PRACTICAS EN LOGO

una determinada acción cuando la condición se cumpla, para continuar después con el proceso general. En el caso de que la condición no se cumpla, no se realiza acción alguna y continúa el proceso.

En este caso, sólo se realiza una determinada acción cuando la condición se cumple, pero sería perfectamente válido realizar también una acción en el caso que no se cumpla.

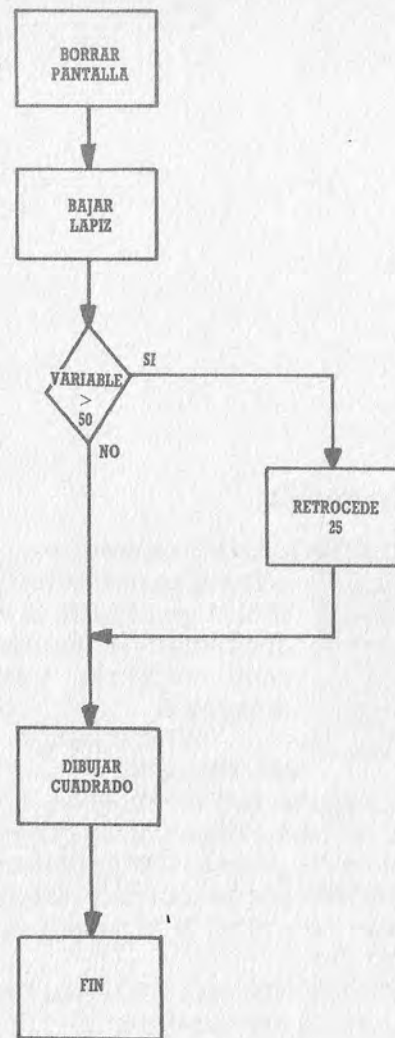
Supongamos, por ejemplo, que en nuestro caso anterior, si observamos que no llueve, tendemos la ropa antes de ir a trabajar.

Nuestro proceso quedaría ahora de la siguiente forma:



Hagamos ahora el diagrama de este proceso:

La orden SI tiene que tener al menos una acción que realizar.



La condición de este proceso es que el valor que introducimos a la variable sea mayor de 50 y la acción a tomar sólo en el caso que se cumpla la condición, es que la Tortuga retroceda 25 pasos antes de dibujar el cuadrado. Si la condición no se cumple, dibujamos el cuadrado exclusivamente.

La forma de interpretarlo dentro de un procedimiento es la siguiente:

En este caso, dependiendo de la condición, realizamos dos acciones diferentes. Una para el caso afirmativo y otra para el caso negativo.

Una condición debe tener por lo menos una acción que realizar y siempre va a tener dos posibles respuestas, CIERTO o FALSO, o

lo que es lo mismo, SI o NO o AFIRMATIVO o NEGATIVO.

La acción a realizar puede estar compuesta a su vez de una o varias acciones.

Supongamos en nuestro ejemplo que la acción a tomar en el caso de que esté lloviendo no es sólo coger el paraguas, sino también coger la gabardina y unos guantes. En este caso realizamos tres acciones cuando la condición establecida sea cierta.

Veamos cómo se puede interpretar en el Logo este tipo de procesos.

Definimos un procedimiento con una variable como dato de entrada para realizar cuadrados de diferentes tamaños. La variable va a determinar la longitud que van a tener los lados del cuadrado. Si la longitud que introducimos, es decir, el valor que le damos a la variable, es mayor de 50, la Tortuga antes de dibujar el cuadrado, deberá retroceder 25 pasos.

Primero veamos las diferentes fases que realiza el proceso:

- 1.º Borrar la pantalla.
- 2.º Bajar el lápiz.
- 3.º Ver si la variable es mayor de 50.
- 4.º Si la variable es mayor de 50, retrocede 25.
- Si no es mayor, no retrocede 25.
- 5.º Dibujar cuadrado.
- 6.º Finalizar.

?PARA CUADRADOS :L

>BP

>BL

>SI :L> 50[RE 25]

>REPITE 4 [AV :L GD 90]

>FIN

La condición la expresamos utilizando la orden SI junto con una expresión que tiene que dar como resultado CIERTO o FALSO.

La condición :L> 50 será cierta cuando el valor de :L sea superior a 50 y falsa cuando :L sea inferior o igual a 50.

Observa que la acción a realizar la escribimos a continuación de la condición y encerrada entre corchetes. De esta forma indicamos que se realice esa acción sólo en el caso que se cumpla la condición.

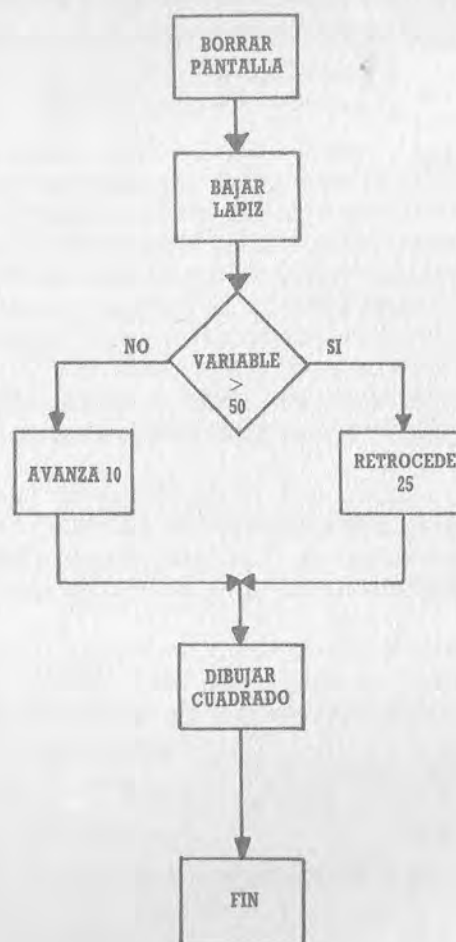
Si la condición no se cumple, no se realiza lo que hay encerrado entre los corchetes,

se pasa directamente a realizar lo que se encuentra en la siguiente línea del procedimiento.

Ampliamos el procedimiento determinando una segunda acción a realizar para el caso que la condición no se cumpla:

Si el valor de la variable no es mayor de 50, la Tortuga deberá avanzar 10 pasos antes de dibujar el cuadrado.

El diagrama del proceso quedaría ahora de la siguiente forma:



En el procedimiento hay que realizar los siguientes cambios:

?PARA CUADRADO :L

>BP

>BL

>SI :L> 50[RE 25] [AV 10]

>REPITE 4 [AV :L GD 90]

>FIN

Con la orden SI el LOGO ya puede tomar decisiones.

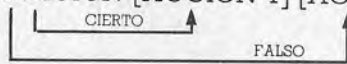
EXPERIENCIAS Y PRACTICAS EN LOGO

Observa que ahora, a continuación de la condición, expresamos las dos acciones a tomar, tanto para el caso afirmativo como para el negativo.

En primer lugar hay que escribir la acción a tomar en el caso que se cumpla la condición y en segundo lugar la que hay que tomar en el caso que no se cumpla. Las dos acciones deben seguir este orden y cada una de ellas tiene que ir encerrada con sus respectivos corchetes.

De forma general podemos expresar:

SI CONDICION [ACCION 1] [ACCION 2]



Si la condición establecida es verdadera (se cumple), se realizan las órdenes que se encuentran en los primeros corchetes y a continuación se realizarán las órdenes que aparecen en la siguiente línea del procedimiento. Si la condición es falsa (no se cumple), no se ejecutarán los primeros corchetes y directamente se realizarán las órdenes que se encuentran en los segundos, para pasar a continuación a realizar también las órdenes que hay en la línea siguiente.

Utilizando dos procedimientos que ya conocemos, uno para dibujar un cuadrado y otro para dibujar un triángulo, vamos a ejecutar dependiendo de una condición uno de ellos.

Establecemos que si la variable A contiene el valor 1 dibujamos un cuadrado, y si tiene el valor 2 dibujamos un triángulo.

```
?PARA CUADRADO
>REPITE 4[AV 40 GD 90]
>FIN
```

```
?PARA TRIANGULO
>REPITE 3[AV 50 GD 120]
>FIN
```

```
?PARA DIBUJO
>HAZ "A 1
>SI :A = 1 [CUADRADO]
>SI :A = 2 [TRIANGULO]
>FIN
```

Si a es igual a 1 llamamos al procedimiento CUADRADO y si es igual a 2 llamamos al procedimiento TRIANGULO.

La orden SI debe escribirse en una misma línea LOGO.

El procedimiento DIBUJO que hemos definido, siempre dibujará un cuadrado, pues A siempre será igual a 1.

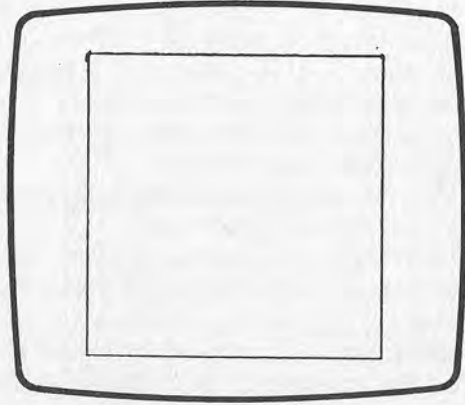


Fig. 1.

Si modificamos la segunda línea del procedimiento DIBUJO

>HAZ "A 1 por >HAZ "A 2

siempre que ejecutemos DIBUJO se dibujará un triángulo.

Una manera de conseguir el dibujo que queramos sin tener que modificar ninguna línea del procedimiento sería definir un procedimiento de una entrada con la que determinar el dibujo a seleccionar.

```
?PARA DIBUJO :A
>SI :A = 1 [CUADRADO]
>SI :A = 2 [TRIANGULO]
>FIN
```

Con este procedimiento que acabamos de definir para dibujar un cuadro teclearemos:

?DIBUJO 1

y para dibujar un triángulo:

?DIBUJO 2

Otra forma de definir el procedimiento anterior, suponiendo que no indicaremos al ejecutarlo valores distinto de 1 y 2, es la siguiente:

```
?PARA DIBUJO :A
>SI :A = 1 [CUADRADO][TRIANGULO]
>FIN
```

Si enviamos al procedimiento DIBUJO el valor 3 obtenemos un triángulo, ya que 3 no es igual a 1 y se ejecutaría el contenido de los siguientes corchetes:

?DIBUJO 3

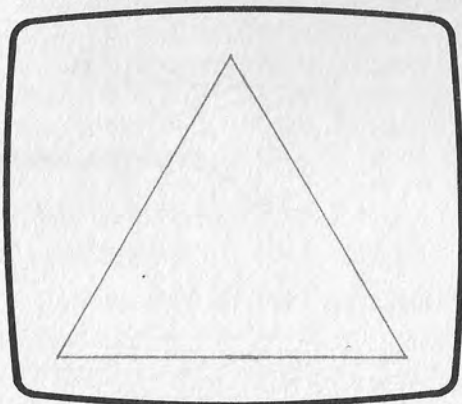


Fig. 2.

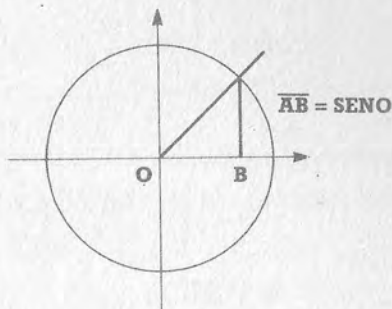
■ Funciones trigonométricas

Las funciones trigonométricas más comunes y que todos conocemos son:

SENO
COSENO
TANGENTE
COTANGENTE
SECANTE
ARCOSENO
ARCOSENO
ARCOTANGENTE

Veamos cómo obtenemos estas funciones con el LOGO, bien utilizando las primitivas que ya vienen implementadas o bien definiendo procedimientos para obtenerlas.

—Función SENO



Dibujo 1.

La obtenemos con la primitiva SEN.

SEN n

Devuelve el seno de n, siendo n un ángulo expresado en grados.

?ESCRIBE SEN 90

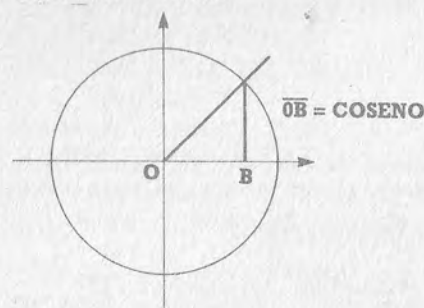
1

?ESCRIBE SEN 30

0.5

En el LOGO del SPECTRUM la primitiva que nos devuelve el seno de un ángulo n es:

—Función COSENO.



Dibujo 2.

La obtenemos con la primitiva COS.

COS n

Devuelve el coseno de un ángulo de n grados.

?ESCRIBE COS 90

0

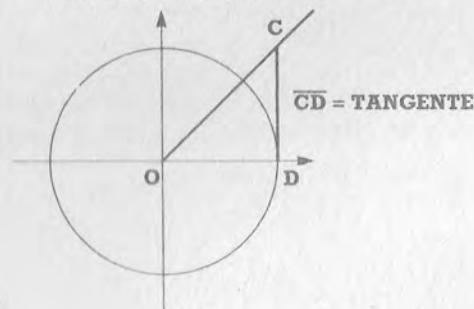
?ESCRIBE COS 60

0.5

En el SPECTRUM la orden para hallar el coseno de un ángulo es:

COSENO n

—Función TANGENTE



Dibujo 3.

Antes de empezar a definir un procedimiento, haz un pequeño estudio del problema.

EXPERIENCIAS Y PRACTICAS EN LOGO

Para obtener esta función definiremos un procedimiento, pues muchas versiones de LOGO carecen de una primitiva para obtenerla.

Sabemos que:

$$\text{TANGENTE } n = (\text{SENO } n) / (\text{COSENO } n)$$

?PARA TANGENTE :N

>HAZ "S SEN :N

>HAZ "C COS :N

>HAZ "T :S / :C

>ESCRIBE :T

>FIN

Hallamos el seno y el coseno de N y dividimos.

?TANGENTE 45

1

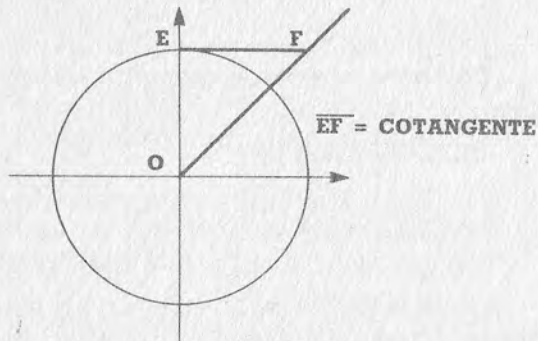
En el LOGO del SPECTRUM sí disponemos de una primitiva para hallar la tangente:

TANGENTE n

?ESCRIBE TANGENTE 60

1.732

—Función COTANGENTE.



Dibujo 4.

Sabemos que:

$$\text{COTANGENTE } n = (\text{COSENO } n) / (\text{SENO } n)$$

Como hemos hecho con la Tangente nos definiremos un procedimiento para obtenerla.

?PARA COTAN :N

>HAZ "S SEN :N

>HAZ "C COS :N

>HAZ "R :C / :S

>ESCRIBE :R

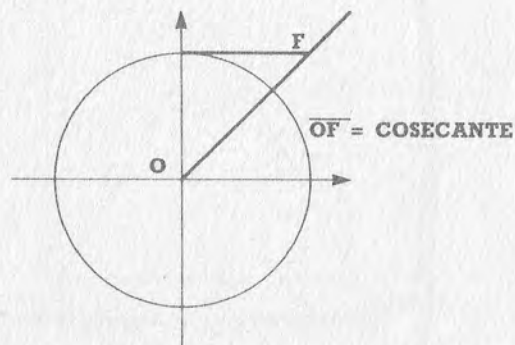
>FIN

Hemos hallado el seno y el coseno de N, luego hemos dividido el coseno entre el seno.

?COTAN 60

0.577

—Función COSECANTE



Dibujo 5

Sabemos que:

$$\text{COSECANTE } n = 1 / (\text{SENO } n)$$

?PARA COSEC :N

>HAZ "S SEN :N

>HAZ "R 1 / :S

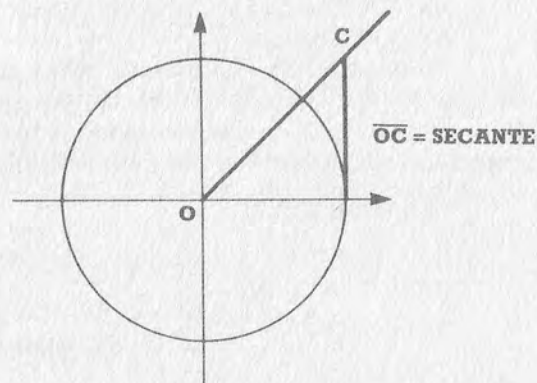
>ESCRIBE :R

>FIN

?COSEC 90

1

—Función SECANTE



Dibujo 6

La secante de un ángulo n es igual:

$$\text{SECANTE } n = 1 / (\text{COSENO } n)$$

?PARA SEC :N

>HAZ "C COS :N

>HAZ "R 1 / :C

Una acción puede estar formada a su vez por varias acciones.

>ESCRIBE :R
>FIN

—Función ARCOTANGENTE m.

Con la función ARCOTANGENTE de m obtenemos el ángulo cuya tangente es m.

Vamos a explicar la función ARCOTANGENTE antes que las funciones ARCOSENO y ARCOCOSENO pues, por lo general, todas las versiones de LOGO disponen de una primitiva para obtener la misma función, pero no para obtener las segundas.

Esta primitiva es:

ARCTAN y

Devuelve el ángulo (en grados) cuya tangente es y.

?ESCRIBE ARCTAN y
?ESCRIBE ARCTAN 1
45

En el LOGO del SPECTRUM la primitiva es:

ARCTG y

—Función ARCOSENO y

Con la función ARCOSENO de y obtenemos el ángulo cuyo seno es y.

Para obtenerla definimos el procedimiento ARCSEN.

?PARA ARCSEN :Y
>HAZ "A :Y / (RC 1 - :Y * :Y)
>ESCRIBE ARCTAN :A
>FIN

Para saber el ángulo cuyo seno es 1 te creamos:

?ARCSEN 0.5
30

—Función ARCOCOSENO y.

Esta función devuelve el arco en grados cuyo coseno es y.

Definimos un procedimiento para obtenerla.

?PARA ARCCOS :Y
>HAZ "A (RC 1 - :Y * :Y) / :Y
>ESCRIBE ARCTAN :A
>FIN

Sea 1 el coseno de un ángulo n. ¿Hallar cuánto vale n?

Para hallar dicho ángulo llamamos al procedimiento que hemos definido indicando el valor de y que ya sabemos que es 1.

?ARCCOS 1
0

NOTA:

En el LOGO del SPECTRUM existen las primitivas ARCSEN, ARCCOS y ARCCOTG, con las que obtendremos el arcoseno, el arco-coseno y el arcocotangente respectivamente.

■ Cuadro resumen

—SI condición [Acción 2]

Dependiendo de la condición que se establece, se efectuarán las órdenes que hay en los primeros corchetes [Acción 1] si la condición es cierta (se cumple), si no es cierta (no se cumple) se realizarán las órdenes que hay en los segundos corchetes [Acción 2].

—SEN n

Devuelve el seno de n, siendo n un ángulo expresado en grados.

—COS n

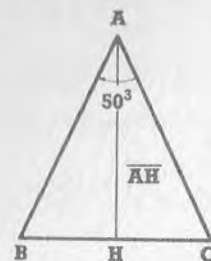
Devuelve el coseno de n, siendo n un ángulo expresado en grados.

—ARCTAN y

Devuelve el arco cuya tangente es y.

■ Ejercicios

1. Define cuatro procedimientos para dibujar cuatro figuras geométricas y establece cuatro condiciones para que dependiendo del valor de una variable dibuje uno u otro.
2. Utilizando las funciones trigonométricas, intenta hallar la altura y el área de un triángulo isósceles, sabiendo que la base del triángulo es de 12fm. y el ángulo opuesto es de 50 grados.



Dibujo 7.

La orden SI tiene dos respuestas: cierto y falso.

EXPERIENCIAS Y PRACTICAS EN LOGO

3. Intenta realizar el siguiente dibujo.

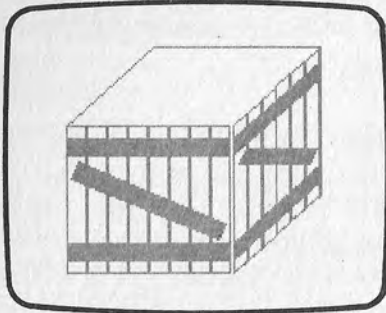


Fig. 3

4. Tomando como base el siguiente display de 7 segmentos, define los procedimientos necesarios para representar los números del 0 al 9, dibujando los segmentos necesarios para cada uno.



Dibujo 8.

Haz los cambios necesarios para que se generen 20 números aleatorios del 0 al 9 y se vayan representando en el display.

5. ¿Son correctas las órdenes?

```

—SI :A = [REPITE 3 [AV 10 GD 120]
—SI :A [AV :A]
—?HAZ "V 10
  ?HAZ "W 20
  ?SI :V < 10[HAZ "V :W]
—?ESCRIBE SIN 45
—?HAZ "V 1
  ?SI :V = (COS 0) [HAZ "V ARCTAN 1]
  ?ESCRIBE :V
—?ESCRIBE 1 / COS 0
    
```

■ Solución a los ejercicios

1. Definidos cuatro procedimientos:

```

?PARA CIRCULO
>REPITE 36 [AV 2 GD 10]
>FIN
    
```

```

?PARA PENTAGONO
>REPITE 5 [AV 20 GD 72]
>FIN
    
```

```

?PARA CUADRADO
>REPITE 4 [AV 30 GD 90]
>FIN
    
```

```

?PARA HEXAGONO
>REPITE 6 [AV 20 GD 60]
>FIN
    
```

Establecemos que:

Si A = 1 ejecutamos CUADRADO
 Si A = 2 ejecutamos PENTAGONO
 Si A = 3 ejecutamos CIRCULO
 Si A = 4 ejecutamos HEXAGONO

```

?PARA DIBUJO :A
>SI A: = 1 [CUADRADO]
>SI A: = 2 [PENTAGONO]
>SI A: = 3 [CIRCULO]
>SI A: = 4 [HEXAGONO]
>FIN
    
```

Una vez definido el procedimiento DIBUJO si queremos dibujar un hexágono lo ejecutaremos:

?DIBUJO 4

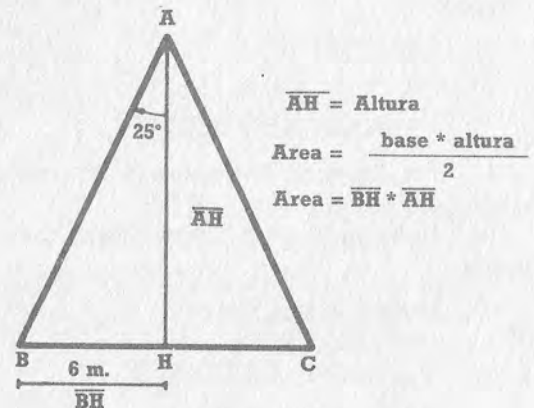
si queremos dibujar un círculo:

?DIBUJO 3

2:

Veamos primero el desarrollo matemático del problema.

Si partimos el triángulo isósceles por la mitad obtenemos dos triángulos rectángulos.



Dibujo 9.

No es necesario indicar la acción a realizar, en el caso de que la condición no se cumpla.

La base de uno de los triángulos rectángulos es la base del isósceles dividida por 2.

$$BH = 6$$

Sabemos que:

$$\text{TANGENTE } 25 = \text{SENO } 25 / \text{COSENO } 25$$

y que

$$\text{tangente } 25 = BH / AH$$

$$\text{tangente } 25 = 6 / AH$$

$$AH = 6 / ((\text{seno } 25) / (\text{coseno } 25))$$

$$AH = 6 / 0,4663$$

$$AH = 12,86 \text{ metros.}$$

Hemos hallado la altura.

Para hallar el área:

$$\text{AREA} = (2 * BH * AH) / 2 = (BH * AH)$$

$$\text{AREA} = 6 * 12,86$$

$$\text{AREA} = 77,20 \text{ metros cuadrados.}$$

Una vez estudiado y comprendido el problema lo escribimos en LOGO.

```
?PARA TRIAN
>HAZ "B 6
>HAZ "T (SEN 25) / (COS 25)
>HAZ "H (B: / T:)
>ESCRIBE :H
>HAZ "S (B: * :H)
>ESCRIBE S
>FIN
```

Cuando ejecutemos este procedimiento primero aparecerá la altura y luego el área.

```
?TRIAN
12,86
77,20
```

3:

Para realizar el dibujo de la caja, definimos tres procedimientos que van a dibujar diferentes partes de ella.

El primero, al que llamamos CAJA, es el que dibuja la estructura de la caja, formada por tablas unidas. De éste, se realiza una llamada a dos procedimientos diferentes que son los que van a dibujar las tablas que atraviesan la caja:

```
?PARA CAJA
>BP
>OT
```

```
>SL
>PONPOS [-44 -44]
>BL
>REPITE 8 [REPITE 2 [AV 8 SL AV 10
BL AV 52 SL AV 10 BL AV 8 GD 90 AV
10 GD 90] GD 90 SL AV 11 GI 90 BL]
>SL
>GD 90 AV 1 GI 90
>BL
>REPITE 6 [REPITE 2 [AV 8 SL AV 10
BL AV 52 SL AV 10 BL AV 8 GD 45 AV
10 GD 135] GD 45 AV 11 GI 45]
>AV 88 GI 90 AV 89
>GI 45 AV 66
>TABLAS1
>TABLAS2
>FIN
```

Si ejecutas este procedimiento en solitario, observarás que vamos dejando un hueco en las tablas para luego dibujar encima de ellas las tablas horizontales.

El procedimiento TABLAS1 dibuja las tablas horizontales que tiene la caja:

```
?PARA TABLAS1
>GI 45 AV 8 GI 90
>REPITE 2 [REPITE 2 [AV 87 GD 90 AV
10 GD 90] AV 87 SL AV 2 BL GI 45 RE-
PITE 2 [AV 66 GD 135 AV 10 GD 45] GD
45 SL AV 3 BL RELLENA SL RE 50 GD
90 AV 3 BL RELLENA SL PONPOS [-44
-26] PONRUMBO 90 BL]
>FIN
```

TABLAS 2 dibuja las tablas inclinadas de la caja:

```
?PARA TABLAS2
>SL
>PONPOS [-38 22]
>PONRUMBO 115
>BL
>REPITE 2 [AV 87 GD 90 AV 10 GD 90]
>PONRUMBO 180
>SL
>AV 2
>PONRUMBO 115
>REPITE 9 [BL RELLENA SL AV 10]
>SL
>PONPOS 90
>PONRUMBO 90
>BL
```

Las acciones tienen que ir encerradas entre corchetes.

EXPERIENCIAS Y PRACTICAS EN LOGO

```
>REPITE 2 [AV 38 GD 115 AV 10 GD 65]
>GD 90
>SL
>AV 8 GI 90 RE 2
>REPITE 7 [BL RELLENA SL AV 7]
>FIN
```

Una vez tenemos definidos los tres procedimientos necesarios, sólo tenemos que ejecutar el primero de ellos para ver el dibujo:

?CAJA

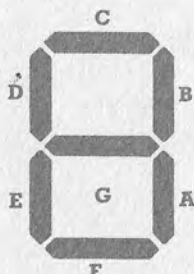
4:

El display tiene siete segmentos y todos ellos son iguales. Lo único que varía de uno a otro es el lugar donde hay que dibujarlo y su sentido.

Vamos a definir en primer lugar un procedimiento para dibujar exclusivamente el segmento.

```
?PARA SEGMENTO
>BL
>AV 20 RE 1
>GD 90 AV 1
>GI 90 RE 17
>GI 90 AV 2
>GD 90 AV 17
>FIN
```

A continuación definimos un procedimiento por cada segmento que posee el display, para dibujar cada uno en su lugar y con su sentido. Primero vamos a designar a cada segmento con una letra:



Dibujo 10.

Definimos cada procedimiento con el nombre de la letra que le hemos dado a cada segmento:

```
?PARA A
>SL
```

```
>PONPOS [0 0]
>PONRUMBO 0
>SEGMENTO
>FIN
```

```
?PARA B
>SL
>PONPOS [0 22]
>PONRUMBO 0
>SEGMENTO
>FIN
```

```
?PARA C
>SL
>PONPOS [-2 44]
>PONRUMBO -90
>SEGMENTO
>FIN
```

```
?PARA D
>SL
>PONPOS [-25 43]
>PONRUMBO 180
>S
>FIN
```

```
?PARA E
>SL
>PONPOS [-25 20]
>PONRUMBO 180
>SEGMENTO
>FIN
```

```
PARA F
>SL
>PONPOS [-23 -1]
>PONRUMBO 90
>SEGMENTO
>FIN
```

```
?PARA G
>SL
>PONPOS [-23 21]
>PONRUMBO 90
>SEGMENTO
>FIN
```

En todos estos procedimientos lo único que hacemos es posicionar a la Tortuga en cada uno de los puntos donde tiene que empezar a dibujar cada segmento y darle la orientación hacia donde tiene que dibujarlo.

Ahora definimos un procedimiento que, por medio de una variable que utilizamos

El ángulo que indicamos junto a una función trigonométrica debe estar expresado en grados.

como dato de entrada, va a ir preguntando cuál es el número introducido. Dependiendo del número, tendrán que dibujarse los segmentos correspondientes para configurarlo en el display.

```
?PARA NUM :N
>BP
>SI :N = 0 [A B C D E F]
>SI :N = 1 [A B]
>SI :N = 2 [C B G E F]
>SI :N = 3 [C B G A F]
>SI :N = 4 [D G B A]
>SI :N = 5 [C D G A F]
>SI :N = 6 [D E F A G]
>SI :N = 7 [A B C]
>SI :N = 8 [A B C D E F G]
>SI :N = 9 [A B C D G]
>FIN
```

Observa que dependiendo del valor de :N, llamamos a los procedimientos que dibujan los segmentos que forman el número.

Para ejecutarlo basta con introducir el nombre de este procedimiento y a continuación el número que queremos que se represente en el display.

Por ejemplo, si queremos obtener el 5:

```
?NUM 5
```

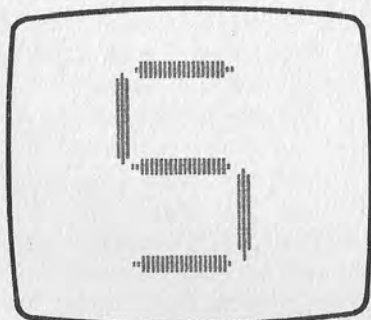


Fig. 4.

Si queremos obtener el tres:

```
?NUM 3
```

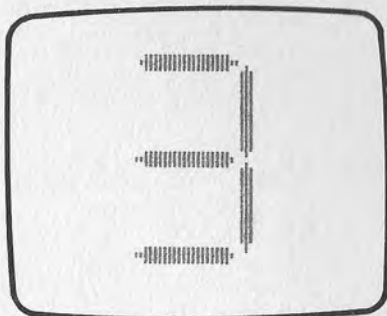


Fig. 5.

A continuación hacemos los cambios necesarios en el procedimiento NUM para conseguir generar un número aleatorio. Para ello, asignamos a la variable :N un número al azar entre 0 y 9. En este caso no se utiliza ninguna variable como dato de entrada.

```
?PARA NUM
>HAZ "N AZAR 10
>SI :N = 0 [A B C D E F]
>SI :N = 1 [A B]
>SI :N = 2 [C B G E F]
>SI :N = 3 [C B G A F]
>SI :N = 4 [D G B A]
>SI :N = 5 [C D G A F]
>SI :N = 6 [D E F A G]
>SI :N = 7 [A B C]
>SI :N = 8 [A B C D E F G]
>SI :N = 9 [A B C D G]
>FIN
```

Como el ejercicio nos pide que generen 20 números aleatorios, definimos otro procedimiento para hacerlo. En él hacemos 20 veces una llamada al procedimiento NUM, realizando antes una limpieza de la pantalla y marcando una pequeña espera antes de volver a repetirlo.

```
?PARA ALEA
>REPITE 20 [LIMPIA NUM ESPERA 10]
>FIN
```

Para ejecutarlo:

```
?ALEA
```

5:

```
—?SI :A = 1 [REPITE 3 [AV 10 GD 120]
```

CORRECTA

Si la variable A es igual a 1, la Tortuga dibujará un triángulo.

```
—?SI :A [AV :A]
```

INCORRECTA

Falta indicar la condición que debe cumplir la variable A.

```
—?HAZ "V 10
```

```
?HAZ "W 20
```

```
?SI :V < 10 [HAZ "V :W]
```

CORRECTA

Asignamos dos valores diferentes a V y W. Si la variable V es menor que 10, se le asigna el valor de W.

La orden SI potencia las posibilidades del Logo.

EXPERIENCIA Y PRACTICAS EN LOGO

Como la variable V es igual a 10, no se realizará ninguna acción.

—?ESCRIBE SIN 45

INCORRECTA

La primitiva que devuelve el seno de un ángulo es SEN no SIN.

El LOGO muestra el mensaje:

NO SE COMO PARA HACER SIN

—HAZ "V 1

?SI :V = (COS 0) [HAZ "V ARCTAN 1]

? ESCRIBE :V

CORRECTA

El coseno de un ángulo de cero grados es 1; por tanto, la condición se cumple y se ejecuta el contenido de los corchetes. El arcotangente de 1 es 45, valor que se le asigna a V.

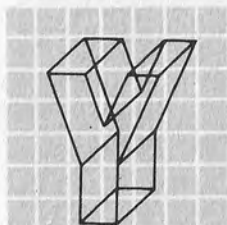
—?ESCRIBE 1 / COS 0

CORRECTA

El coseno del ángulo cero es 1.
La orden anterior devuelve 1.

- ***Si la condición no se cumple y no existe acción para este caso, se ejecutan las órdenes que haya en la línea siguiente.***
- ***Las acciones pueden contener llamadas a otros procedimientos.***
- ***Las condiciones pueden ser de relación y de consulta.***

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



A hemos visto cómo realizar el movimiento de figuras sencillas y de figuras algo más complejas. También vimos cómo controlar el movimiento desde el teclado y cómo simular movimientos simultáneos dentro de nuestro

ordenador. Sabemos cómo hacer que ciertos objetos reboten contra las paredes de la pantalla o contra otros objetos. Hemos aprendido cómo realizar movimientos animados de una o varias figuras a la vez por la pantalla. Lo único que nos queda por ver sobre el movimiento y la animación es hablar de algunos efectos o movimientos especiales que no se pueden encuadrar bajo ninguna denominación específica. Tal es el caso de la simulación de explosiones, choques, realización de disparos, etc. A todos estos tipos de efectos que aún están relacionados algunos de ellos con el movimiento no lo son, les llamaremos efectos especiales y los veremos en este tomo.

Efectos especiales

Como hemos dicho un poco más arriba, dentro de este grupo vamos a incluir efectos como:

- Explosión de una nave o cualquier otro objeto.
- Choques de unos objetos contra otros.
- Disparos hechos por nosotros o por nuestros enemigos.

Los dos primeros programas que vamos a ver nos muestran cómo un pistolero (en el SPECTRUM) y un tanque (en lo demás ordenadores) disparan una bala para derribar unos cuadrados que hay esparcidos en la pantalla. Estos dos programas son el número uno y el número dos. El primero es sólo válido para los usuarios del SPECTRUM y el segundo para el resto de los ordenadores.

```
100 REM *****
110 REM * EL LLANERO SOLITARIO *
120 REM *****
130 REM
140 REM *** DEFINICION DEL LLANERO SOLITARIO ***
150 REM
160 DATA 0,0,60,60,255,60,62,60
170 DATA 24,60,126,127,127,126,126,126
180 DATA 0,0,0,0,128,254,120,0
190 DATA 126,126,102,102,102,102,102,119
200 REM
210 REM *** LECTURA DE LAS DATAS ***
220 REM
230 FOR I=0 TO 31
240   READ N
250   POKE USR "A"+I,N
260 NEXT I
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

270 REM
280 REM *** COLOCACION DE LOS CUADRADOS ***
290 REM
300 INK 0
310 BORDER 1
320 PAPER 8
330 CLS
340 FOR I=1 TO 5
350   LET C=INT(RND*5)+1
360   LET XT=INT(RND*20)+1
370   LET YT=INT(RND*16)+15
380   PRINT INK C;AT XT,YT;CHR$ 143
390 NEXT I
400 LET X=11
410 LET Y=1
420 LET D=0
430 PRINT AT X-1,Y;CHR$ 144
440 PRINT AT X,Y;CHR$ 145;CHR$ 146
450 PRINT AT X+1,Y;CHR$ 147
460 LET A$=INKEY$
470 IF A$="" THEN GOTO 460
480 IF CODE A$<>54 AND CODE A$<>55 AND CODE A$<>56 THEN GOTO 460
490 IF CODE A$<>56 THEN PRINT AT X-1,Y;" ";AT X,Y;" ";AT X+1,Y;" "
500 GOSUB VAL(A$)*100
510 PAUSE 5
520 GOTO 430
600 REM
610 REM *** MOVIMIENTO HACIA ABAJO ***
620 REM
630 LET X=X+1
640 IF X>20 THEN LET X=20
650 RETURN
700 REM
710 REM *** MOVIMIENTO HACIA ARRIBA ***
720 REM
730 LET X=X-1
740 IF X<1 THEN LET X=1
750 RETURN
800 REM
810 REM *** DISPARO DEL LLANERO ***
820 REM
830 IF D=5 THEN GOTO 910
840 LET YB=Y+2
850 PRINT AT X,YB;" ."
860 LET YB=YB+1
870 IF YB>30 THEN PRINT AT X,YB;" ":LET D=D+1:GOTO 890
880 GOTO 850
890 RETURN
900 REM
910 REM *** FIN DEL JUEGO ***
920 REM
930 PRINT AT 0,0;"EL JUEGO HA TERMINADO"
940 PRINT "ADIOS ...."
950 STOP

```

Este primer programa, válido sólo para el SPECTRUM, nos permitirá mover al LLANERO SOLITARIO arriba y abajo por la pantalla y disparar a los cuadros de colores que han aparecido en ella. El juego no es nada del otro mundo, pero nos permite ver cómo realizar rutinas de disparo. Las teclas que se han de uti-

lizar para mover al pistolero por la pantalla y para disparar son las siguientes:

- 6 Movimiento hacia arriba.
- 7 Movimiento hacia abajo.
- 8 Disparar.

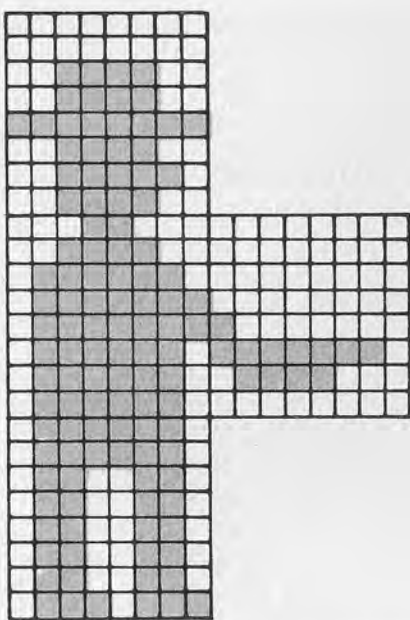


Fig. 1. Definición del Llanero solitario en el SPECTRUM.

Hay que decir que sólo disponemos de cinco disparos para deshacernos de los cinco cuadraditos que hay en la pantalla.

El funcionamiento del programa es el siguiente:

Líneas 160 a 190. En estas líneas se encuentran las líneas de DATA que tienen almacenados los números que definen la figura del pistolero.

Línea 230. Comenzamos un bucle 0 y 31 dentro del cual vamos a leer los números de las DATAS uno a uno y los vamos a ir POKEando en la zona de memoria donde se encuentran los caracteres definidos por el usuario.

Línea 240. Leemos los valores de las líneas DATA y los almacenamos uno a uno en la variable N.

Línea 250. Pokeamos el valor que acabamos de leer de las líneas DATA.

Línea 260. Aquí termina el bucle que se encarga de definir los caracteres que darán forma al pistolero.

Líneas 300 a 330. Ponemos la tinta de color negro, el papel de color amarillo, y el borde de color azul oscuro. Borrarnos la pantalla.

Línea 340. Comenzamos un bucle desde 1 hasta 5 dentro del cual vamos a dibujar cinco cuadraditos en unas posiciones aleatorias de la pantalla.

Línea 350. Hacemos que la variable C tenga un valor aleatorio entre 1 y 5. Esta va-

riable definirá el color del cuadrado que se imprimirá a continuación.

Líneas 360 y 370. Gracias a la función RND obtenemos dos números que serán las coordenadas en la pantalla del cuadrado que vamos a dibujar a continuación. El rango de la coordenada en X va desde la primera fila a la número 20. El rango de la coordenada en Y va desde la columna 15 hasta la 30.

Línea 380. Imprimimos un cuadrado de color XC en la posición de pantalla dada por las coordenadas XT, YT.

Línea 390. Aquí termina el bucle que se encarga de dibujar cinco cuadrados en la pantalla.

Línea 400. Asignamos a la variable X el valor 11. Esta será la fila donde aparecerá el Llanero al principio del programa.

Línea 410. Asignamos a la variable Y el valor 1. Esta será la columna donde colocaremos y por donde se moverá el pistolero.

Línea 420. Ponemos el contador de disparos realizados (D) a cero.

Línea 430. Imprimimos la cabeza del pistolero.

Línea 440. Imprimimos el cuerpo y la mano con la pistola.

Línea 450. Imprimimos las piernas.

Línea 460. Miramos si el usuario ha pulsado alguna tecla.

Línea 470. Si el usuario no pulsó ninguna tecla entonces nos volvemos a la línea 460 para volver a preguntar. Este bucle va a repetirse hasta que se pulse alguna tecla.

Línea 480. Comprobamos que la tecla pulsada está dentro de las permitidas. Estas son el 6, el 7 y el 8. En caso de que no sea una de las permitidas se vuelve a la línea 460.

Línea 490. Si el usuario pulsó la tecla de movimiento hacia arriba o hacia abajo (no pulsó la tecla de disparo), entonces borramos al pistolero de la pantalla. Más adelante lo volveremos a imprimir en la nueva posición.

Línea 500. Transferimos el control a la línea 600 o a la 700 o a la 800 dependiendo de la tecla pulsada por el usuario.

Línea 510. Una vez que se ha ejecutado la rutina dependiendo de la acción que quiso realizar el usuario y que se devolvió el control al programa principal, se hace una pequeña pausa antes de leer la siguiente pulsación del teclado.

Línea 520. Nos volvemos a la línea 430. En esta línea vamos a imprimir de nuevo al pistolero. Si se varió la posición de éste, apa-

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

recerá en la nueva posición. Si no variamos su posición (sólo pulsamos la tecla de disparo) entonces aunque también imprimiremos al pistolero, no lo notaremos, puesto que ya estaba impreso.

Línea 630. Si el usuario pulsó la tecla de movimiento hacia abajo, hacemos que la variable X se incremente en uno.

Línea 640. Si el pistolero se encuentra en una línea mayor que la 20 entonces hacemos que X sea igual a 20. Con esto conseguimos que el pistolero no intente salirse por la parte inferior de la pantalla y que el ordenador no nos dé error.

Línea 650. Devolvemos el control al programa principal para dibujar al Llanero en su nueva posición.

Línea 730. Si el usuario pulsó la tecla de movimiento hacia arriba entonces decremamos la variable numérica X en uno.

Línea 740. Si el Llanero se encuentra en una línea menor que la primera entonces, para que el ordenador no nos dé error al intentar imprimir al pistolero fuera de la pantalla, hacemos que X sea igual a 1.

Línea 750. Devolvemos el control al programa principal para que imprima al pistolero en su nueva posición.

Línea 830. Llegamos a esta línea cuando el usuario ha pulsado la tecla de disparo. Esta es la parte más importante del programa, pues a ella está dedicado este tomo. En esta línea preguntamos si el pistolero ha gastado todas sus balas. En caso afirmativo nos vamos a la línea 910 para darle un mensaje de despedida.

Línea 840. Hacemos que la variable YB tome el valor de la variable Y+2. Esto sirve para indicar desde qué posición de la pantalla va a salir el disparo (la bala) del pistolero.

Línea 850. Imprimimos en dicha posición un espacio en blanco seguido de un punto que actuará como bala.

Línea 860. Incrementamos el valor de YB en uno. Esto es necesario porque, como el movimiento va a ser de izquierda a derecha, tenemos que aumentar el número de columna donde tendremos que imprimir dentro de un momento la bala que hemos disparado.

Línea 870. Preguntamos si la bala ya ha llegado a la columna número 30. En caso afirmativo imprimimos en la posición donde se encuentra dibujada un espacio en blanco, sumamos uno al contador de balas gastadas y nos vamos a la línea 890 para volver al programa principal.

Línea 880. Si todavía no hemos llegado al límite de la pantalla entonces nos volvemos a la línea 850 para imprimir la bala en su nueva posición y calcular la siguiente.

Línea 890. Devolvemos el control al programa principal.

Líneas 930 a 950. Sacamos un mensaje de despedida y le decimos al ordenador que pare.

A continuación vamos a ver un programa muy parecido al anterior pero en versión para IBM. Este programa nos permite mover un tanque hacia arriba y hacia abajo por la pantalla y disparar a una serie de cuadrados que aparezcan en pantalla.

Las teclas que nos permitirán controlar el movimiento del tanque y disparar son las siguientes:

- P Movimiento hacia arriba.
- L Movimiento hacia abajo.
- Q Disparar.

```
100 REM *****
110 REM * ATAQUE BELICO *
120 REM *****
130 REM
140 CLS
150 RANDOMIZE TIMER
160 REM
170 REM *** SITUACION DE LOS BLANCOS A DISPARAR ***
180 REM
190 FOR I=1 TO 5
200   LET C=INT(RND*16)
210   IF C=2 THEN GOTO 200
220   LET XT=INT(RND*18)+2
230   LET YT=INT(RND*20)+20
240   COLOR C
```

```

250 LOCATE XT,YT
260 PRINT CHR$(177)
270 NEXT I
280 REM
290 REM *** DISEÑO DEL TANQUE ***
300 REM
310 LET X=11
320 LET Y=2
330 LET D=0
340 COLOR 4
350 LOCATE X-2,Y+4
360 PRINT CHR$(218);CHR$(196)
370 LOCATE X-1,Y+4
380 PRINT CHR$(219);CHR$(219);STRING$(3,205)
390 LOCATE X,Y+1
400 PRINT STRING$(8,219)
410 LOCATE X+1,Y
420 PRINT STRING$(10,219)
430 REM
440 REM *** MOVIMIENTO Y DISPARO ***
450 REM
460 LET A$=INKEY$
470 IF A$="" THEN GOTO 460
480 IF A$<>"P" AND A$<>"L" AND A$<>"Q" THEN GOTO 460
490 IF A$="Q" THEN GOTO 580
500 LOCATE X-2,Y+4
510 PRINT " "
520 LOCATE X-1,Y+4
530 PRINT SPC(5)
540 LOCATE X,Y+1
550 PRINT SPC(8)
560 LOCATE X+1,Y
570 PRINT SPC(10)
580 IF A$="Q" THEN GOSUB 630
590 IF A$="L" THEN GOSUB 770
600 IF A$="P" THEN GOSUB 830
610 GOTO 350
620 REM
630 REM *** DISPARO DEL TANQUE ***
640 REM
650 IF D=5 THEN GOTO 890
660 LET YB=Y+9
670 LOCATE X-1,YB
680 PRINT " ";CHR$(249)
690 LET YB=YB+1
700 IF YB>38 THEN LOCATE X-1,YB:PRINT " ":GOTO 740
710 FOR I=1 TO 50
720 NEXT I
730 GOTO 670
740 LET D=D+1
750 RETURN
760 REM
770 REM *** MOVIMIENTO HACIA ABAJO ***
780 REM
790 LET X=X+1
800 IF X>20 THEN LET X=20
810 RETURN
820 REM
830 REM *** MOVIMIENTO HACIA ARRIBA ***
840 REM
850 LET X=X-1
860 IF X<3 THEN LET X=3
870 RETURN
880 REM
890 REM *** FIN DEL JUEGO ***

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

900 REM
910 LOCATE 1,1
920 PRINT "EL JUEGO HA TERMINADO."
930 PRINT "ADIOS ...."
940 END

```

Como el programa está hecho para el IBM, para que funcione en otros ordenadores habrá que realizar las siguientes modificaciones:

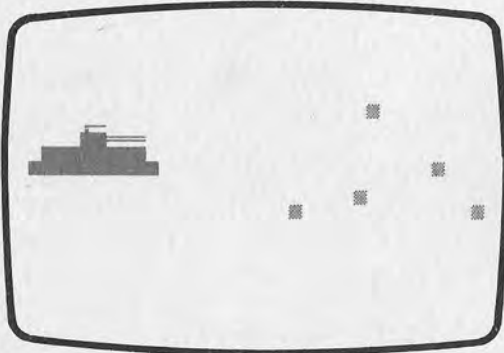


Fig. 2. El programa 2 en ejecución.

COMMODORE:

```

140 PRINT CHR$(147)
150 LET A=RND(-TI)
200 REM
210 REM
220 LET XT=INT(RND(1)*18)+2
230 LET YT=INT(RND(1)*20)+19
240 REM
250 POKE 214,XT:POKE 211,YT
250 PRINT CHR$(166)
340 REM
350 POKE 214,X-2:POKE 211,Y+4
360 PRINT CHR$(176);CHR$(99)
370 POKE 214,X-1:POKE 211,Y+4
380 ?CHR$(166);CHR$(166);CHR$(168);CHR$(168);CHR$(168)
390 POKE 214,X:POKE 211,Y+1
400 FOR Z=1 TO 8:PRINT CHR$(166);:NEXT Z
410 POKE 214,X+1:POKE 211,Y
420 FOR Z=1 TO 10:PRINT CHR$(166);:NEXT Z
460 GET A$
500 POKE 214,X-2:POKE 211,Y+4
520 POKE 214,X-1:POKE 211,Y+4
530 PRINT " ":REM 3 ESPACIOS
540 POKE 214,X:POKE 211,Y+1
550 PRINT " ":REM 8 ESPACIOS
560 POKE 214,X+1:POKE 211,Y
570 PRINT " ":REM 10 ESPACIOS
670 POKE 214,X-1:POKE 211,YB
680 PRINT " "
700 IF YB>37 THEN POKE 214,X-1:POKE 211,YB:PRINT
" ":GOTO 740
910 POKE 214,0:POKE 211,0

```

AMSTRAD:

```

150 RANDOMIZE TIME
200 LET C=INT(RND(1)*7)+1
220 LET XT=INT(RND(1)*18)+2
230 LET YT=INT(RND(1)*20)+20
240 INK C
250 LOCATE YT,XT
260 PRINT CHR$(127)
340 INK 4
350 LOCATE Y+4,X-2
360 PRINT CHR$(150);CHR$(154)
370 LOCATE Y+4,X-1
380 PRINT CHR$(143);CHR$(143);STRING$(3,140)
390 LOCATE Y+1,X
400 PRINT STRING$(8,143)
410 LOCATE Y,X+1
420 PRINT STRING$(10,143)
500 LOCATE Y+4,X-2
520 LOCATE Y+4,X-1
540 LOCATE Y+1,X
560 LOCATE Y,X+1
670 LOCATE YB,X-1
680 PRINT " ";CHR$(144)
700 IF YB>38 THEN LOCATE YB,X-1:PRINT " ":GOTO 740

```

MSX:

```

150 LET A=RND(-TIME)
200 LET C=INT(RND(1)*7)+1
220 LET XT=INT(RND(1)*18)+2
230 LET YT=INT(RND(1)*20)+20
250 LOCATE YT,XT
260 PRINT CHR$(215)
350 LOCATE Y+4,X-2
360 PRINT CHR$(199);CHR$(195)
370 LOCATE Y+4,X-1
380 PRINT CHR$(219);CHR$(219);STRING$(3,195)
390 LOCATE Y+1,X
400 PRINT STRING$(8,219)
410 LOCATE Y,X+1
420 PRINT STRING$(10,219)
500 LOCATE Y+4,X-2
520 LOCATE Y+4,X-1
540 LOCATE Y+1,X
560 LOCATE Y,X+1
670 LOCATE YB,X-1
680 PRINT " ";CHR$(188)
700 IF YB>38 THEN LOCATE YB,X-1:PRINT " ":GOTO 740

```

Una vez vistas las modificaciones vamos a ver el funcionamiento del programa línea a línea.

Línea 140. Borrarnos la pantalla.

Línea 150. Inicializamos el generador de números aleatorios utilizando el valor del reloj que tiene el ordenador.

Línea 190. Comenzamos un bucle dentro del cual vamos a imprimir cinco cuadrados de colores dentro de la pantalla.

Línea 200. Obtenemos un número (C) entre 1 y 16 (en el IBM) o entre 1 y 7 (en los demás ordenadores) que nos dirá de qué color va a ser el cuadrado que tenemos que imprimir.

Línea 220. Obtenemos otro número (XT) que nos dará la coordenada en Y de la pantalla donde se colocará el cuadro que vamos a imprimir a continuación.

Línea 230. Obtenemos la coordenada en X (YT) de dicho cuadrado.

Línea 240. Le decimos al ordenador que queremos imprimir con el color C. En el COMMODORE esta instrucción no se ha puesto porque es algo más complicada y porque es mejor dejarla sin explotar hasta que lleguemos al capítulo del color.

Línea 250. Colocamos el cursor en la posición de la pantalla que acabamos de hallar.

Línea 260. E imprimimos un cuadrado de color C.

Línea 270. Aquí termina el bucle que se encarga de imprimir los cinco blancos que tendrá que derribar el tanque.

Línea 310. Asignamos a la variable X el valor 11. Esta variable nos dice en qué fila se va a colocar el tanque al empezar a ejecutarse el programa.

Línea 320. Asignamos a la variable Y el valor 2. Esta variable almacenará la columna donde empieza la figura del tanque.

Línea 330. La variable D contará el número de disparos que hemos hecho. Estos tienen un límite de 5. En esta línea ponemos a cero esta variable.

Línea 340. Ponemos el color que tendrá el tanque y los disparos que éste realice.

Líneas 350 a 420. Utilizando los caracteres semigráficos del ordenador dibujamos un tanque.

Línea 460. Miramos si el usuario ha pulsado alguna tecla.

Línea 470. Si no pulsó ninguna nos volvemos a la línea anterior para volver a preguntar. Este bucle se repite hasta que el usuario pulsa alguna tecla, sea o no sea permitida.

Línea 480. Si la tecla que ha pulsado no es ni una P ni una Q entonces volvemos a la línea 460 para esperar a que pulse algunas de estas teclas. Como se puede apreciar, sólo se pueden pulsar teclas que estén en mayúsculas.

Línea 490. Si la tecla pulsada fue la de disparo nos vamos a la línea 580. Esta línea está puesta, pues si el usuario pulsó otra tecla (que tiene que ser de movimiento) vamos a borrar el tanque de su posición actual para más tarde volverlo a imprimir en su nueva posición. Como cuando se dispara no hay variación en su posición, nos tenemos que saltar el grupo de líneas que vienen a continuación.

Líneas 500 a 570. Borrarnos el tanque de su posición actual.

Línea 580. Si el usuario pulsó la tecla Q entonces nos vamos a la línea 630 para gestionar el disparo.

Línea 590. Si el usuario pulsó la tecla L entonces nos vamos a la línea 770 para gestionar el movimiento hacia abajo.

Línea 600. Si el usuario pulsó la tecla P nos vamos a la línea 830.

Línea 610. Una vez que hemos vuelto de la rutina correspondiente nos volvemos a la línea 350 para recoger la siguiente pulsación del usuario.

Línea 650. Llegamos a esta línea cuando el usuario ha pulsado la tecla Q. Primero preguntamos si ya ha hecho cinco disparos. En caso afirmativo nos vamos a la línea 890 para imprimir un mensaje de despedida.

Línea 660. Si todavía no ha agotado las balas que tenía el tanque, entonces hacemos que la variable YB tome el valor de la posición desde donde saldrá la bala que queremos lanzar.

Línea 670. Colocamos el cursor en dicha posición.

Línea 680. E imprimimos un carácter en blanco seguido del carácter semigráfico que utilizaremos como bala.

Línea 690. Hacemos que la posición de la bala aumente en uno ($YB=YB+1$) para poderla colocar, en la siguiente vuelta, una posición más a la derecha de la que tenía anteriormente.

Línea 700. Si la bala está en la columna 39 ($YB>38$) entonces tenemos que hacer que desaparezca. Por ello colocamos el cursor en la posición pertinente e imprimimos dos espacios. A continuación nos vamos a la línea 740 que nos permitirá volver al programa principal.

Líneas 710 y 720. Realizamos un bucle de retardo para que la bala no se mueva con excesiva rapidez. Variando el número que va después de la palabra TO podremos hacer que la bala vaya más de prisa o más despacio.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Línea 730. Decimos al ordenador que continúe el programa en la línea 670 para imprimir la bala una posición más hacia la derecha y continuar con el movimiento de ésta.

Línea 740. Como acabamos de hacer un disparo, hacemos que la variable encargada de contarlos (D) aumente en uno.

Línea 750. Devolvemos el control al programa principal.

Línea 790. Aparecemos en esta línea cuando nos queremos mover hacia abajo. Por ello incrementamos la variable X en uno.

Línea 800. Si X es mayor que 20 significa que estamos a punto de tocar el borde inferior de la pantalla. Hacemos que X sea igual a 20 para que el tanque no se salga de la pantalla y para que el ordenador no nos dé un error.

Línea 810. Nos volvemos al programa principal. Una vez en éste, imprimiremos el tanque en su nueva posición.

Línea 850. Si el usuario pulsó la tecla para mover el tanque hacia arriba, aparecemos en esta línea. Decrementamos el valor de X para disminuir el número de fila.

Línea 860. Si la variable X es menor de 3 entonces el tanque se va a salir por la parte superior de la pantalla. Para remediarlo hacemos que X sea igual a 3 y así el ordenador no nos da ningún mensaje de error al intentar imprimir el tanque fuera de la pantalla.

Línea 870. Devolvemos el control al programa principal. Una vez allí imprimiremos el tanque en su nueva posición.

Líneas 910 a 940. Imprimimos un mensaje de despedida y terminamos el programa.

Como se puede apreciar, mientras el tanque (o el Llanero solitario) está disparando, no podemos movernos. Hacer un programa que nos permita movernos a la vez que un disparo que nosotros hemos hecho siga su curso entra dentro de lo que hemos llamado movimientos simultáneos. En el próximo tomo veremos unas modificaciones de estos programas que nos permitirán, no sólo movernos mientras la bala sigue su curso, sino detectar cuándo ésta ha chocado con algo. También veremos cómo realizar explosiones y colisiones al chocar la bala con cualquier objeto.

Se insta al lector para que intente resolver este problema antes de ver la solución. Se pide un programa como el que hemos visto pero con las siguientes modificaciones:

- Mientras la bala está moviéndose, el tanque (o pistolero) también se puede y debe mover.

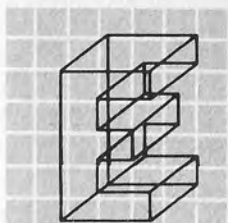
- Cuando la bala choque con un cuadrado tiene que haber una explosión.

- Si la bala no choca con ningún bloque en su trayectoria, invertirá el sentido de su movimiento e intentará matar al tanque (o pistolero).

- Si la bala choca con el tanque (o pistolero), éste tiene que deshacerse en trozos.

Si realizamos un programa así, podremos decir que hemos terminado con los capítulos referentes al movimiento.

TRUCOS Y RUTINAS BASICAS



N el tomo anterior ya vimos algunas rutinas de ordenación numérica y alfanumérica. En este tomo veremos unos cuantos más y terminaremos con este tipo de rutina.

Las rutinas que vienen a continuación pueden ser utilizadas con el programa número uno del tomo anterior. Todas, al igual que las que ya hemos visto, se encuentran localizadas a partir de la línea 100.

El primero de los programas realiza un SORT (una ordenación) de una lista de elementos por el método de cambio de tabla. Este método consiste en crear dos tablas distintas. En una de ellas estarán almacenados los elementos a ordenar. En la otra iremos almacenando los elementos una vez ordenados. Se recomienda la utilización de este método para listas que estén parcialmente ordenadas. De todas maneras, su velocidad es bastante aceptable.

```
1000 REM *****
1010 REM *
1020 REM * <<< ORDENACION POR EL METODO DE CAMBIO DE TABLA >>> *
1030 REM *
1040 REM * VARIABLES QUE HAY QUE PASARLE AL PROGRAMA
1050 REM * -----
1060 REM *
1070 REM * NU = NUMERO DE ELEMENTOS A ORDENAR
1080 REM * T$( ) = TABLA CON LOS ELEMENTOS A ORDENAR
1090 REM *
1100 REM * VARIABLES INTERNAS USADAS POR LA RUTINA
1110 REM * -----
1120 REM *
1130 REM * A$ = VARIABLE DE INTERCAMBIO
1140 REM * I = VARIABLE INDICE DEL PRIMER BUCLE
1150 REM * J = VARIABLE INDICE DEL SEGUNDO BUCLE
1160 REM * R$( ) = TABLA AUXILIAR
1170 REM *
1180 REM * LA SUBROUTINA NOS RETORNA CON LOS DATOS
1190 REM * ORDENADOS EN T$( ).
1200 REM *
1210 REM *****
1220 REM
1230 DIM R$(NU)
1240 LET R$(1)=T$(1)
1250 FOR I=2 TO NU
1260 LET R$(I)=T$(I)
1270 IF R$(I)>=R$(I-1) THEN GOTO 1290
1280 GOSUB 2000
1290 NEXT I
1300 FOR I=1 TO NU
1310 LET T$(I)=R$(I)
1320 NEXT I
1330 RETURN
2000 REM
```

```

2010 REM *****
2020 REM * SUBROUTINA DE INTERCAMBIO *
2030 REM *****
2040 REM
2050 FOR J=I TO 2 STEP -1
2060   IF R$(J)>R$(J-1) THEN RETURN
2070   LET A$=R$(J)
2080   LET R$(J)=R$(J-1)
2090   LET R$(J-1)=A$
2100 NEXT J
2110 RETURN

```

El programa puede funcionar en todos los ordenadores. Si queremos utilizarlo en el SPECTRUM tendremos que variar la línea 1230 por:

```
1230 DIM R$(NU,100)
```

El segundo número dentro de la dimensión (100) le dice al ordenador la longitud máxima de cada elemento. Así, si necesitamos que dicha longitud sea mayor, sólo tenemos que variar esta línea. Por ejemplo, si cada elemento puede tener hasta 135 caracteres de longitud, entonces pondremos:

```
1230 DIM R$(NU,135)
```

También sería conveniente que variásemos la longitud cuando ésta sea menor de 100 para ahorrar memoria.

La explicación del funcionamiento de la rutina línea a línea es el siguiente:

Línea 1230. Dimensionamos la matriz R\$() para que tenga el número de elementos que queremos ordenar.

Línea 1240. Asignamos al primer elemento de la tabla R\$() el primer elemento de T\$(). Esto se hace necesario porque más adelante, cuando empecemos a comparar, vamos a necesitar que en la primera posición de R\$() haya algo.

Línea 1250. Comenzamos un bucle entre 2 y el número de elementos a ordenar dentro del cual se producirá la ordenación.

Línea 1260. Asignamos a R\$(i) el valor de T\$(i). Lo que hace esta línea es ir pasando los elementos que se encuentran en la tabla T\$() a R\$().

Línea 1270. Si el número que acabamos de almacenar en R\$() es mayor o igual que el que está almacenado en la posición anterior, entonces nos vamos a la línea 1290 para continuar con el siguiente elemento. Lo que hace esta línea es mirar si el elemento que acabamos de introducir ha quedado ordenado o no. Como los anteriores elementos de la tabla van

siendo ordenados en cada paso, si el elemento que acabamos de colocar en ésta, es mayor o igual que el anterior, significa que ya está ordenado.

Línea 1280. En caso negativo, si el elemento que acabamos de introducir es menor que el anterior, lo cual significa que tiene que estar por lo menos antes que éste, nos vamos a la línea 2000 para colocar el nuevo elemento donde corresponda.

Línea 1290. Y terminamos el bucle. Una vez llegados a este punto tenemos toda la tabla ordenada en R\$().

Línea 1300. Comenzamos un nuevo bucle dentro del cual vamos a pasar todos los elementos, que se encuentran ordenados en R\$(), a T\$().

Línea 1310. Los pasamos uno a uno.

Línea 1320. Terminamos el bucle. Hay que decir que si el lector quiere tener una tabla con los elementos ordenados, pero también le interesa, por el motivo que sea, tener otra tabla con los elementos tal y como estaban antes de sufrir la ordenación, puede quitar estas últimas tres líneas. Con ello se consigue que tengamos la tabla ordenada en R\$() y la tabla tal y como era antes de la ordenación en T\$().

Línea 1330. Devolvemos el control al programa principal y que llamó a esta rutina.

A continuación viene la rutina que se encarga de colocar el nuevo elemento que acabamos de introducir en R\$() en el lugar que le corresponda.

Línea 2050. Empieza un bucle dentro del cual vamos a ordenar el elemento que acabamos de introducir. El bucle va desde I hasta 2 (con incremento negativo) para empezar la ordenación desde el elemento que acabamos de introducir. El por qué empezamos por el elemento que acabamos de introducir lo vemos en la línea siguiente.

Línea 2060. Preguntamos si el elemento al cual apunta la variable J (variable índice

del bucle) es mayor que el anterior. En caso afirmativo nos volvemos a la rutina principal. Esta comprobación nos sirva para saber cuándo está ordenada totalmente la parte de la tabla en la que estamos trabajando, pues, como ésta estaba ordenada ya de la última pasada, y lo único que queríamos hacer era colocar el último elemento en la posición que le correspondía, cuando éste esté colocado, todo el resto estará ordenado.

Línea 2070. Hacemos que A\$ sea igual a R\$(j). Esta variable (A\$) la utilizaremos para realizar el intercambio de posición de dos elementos de la tabla.

Línea 2080. Hacemos que R\$(j) sea igual a R\$(j-1).

Línea 2090. Pasamos el valor de A\$ al elemento R\$(j-1). Una vez que estas tres líneas han sido ejecutadas, los dos elementos sobre los que trabaja están ordenados.

Línea 2100. Aquí termina el bucle.

Línea 2110. Nos volvemos a la rutina principal.

Se recuerda al lector que para utilizar estas rutinas puede utilizar el programa núme-

ro uno del tomo anterior y ver, así, cómo funcionan.

El segundo programa que vamos a ver también realiza la ordenación de todos los elementos utilizando una segunda tabla auxiliar. El método no es muy bueno ni muy rápido si la tabla a ordenar no está parcialmente ordenada. Por otra parte, la tabla origen desordenada se pierde, ya que cada vez que cogemos un elemento de ésta, se pone en su lugar el carácter 255 para saber que ese elemento ya ha sido ordenado en la tabla auxiliar. Debido a esto, no podemos tener por un lado la tabla ordenada y por el otro la tabla antes de sufrir la ordenación.

Este programa sólo aparece a título de ejemplo y para que se sepa de su existencia, pues se utiliza en muy pocas ocasiones.

El programa ordena por la búsqueda del mínimo. Esto consiste en encontrar el elemento de la tabla origen T\$() más pequeño y ponerlo en la tabla destino R\$(). Cada vez que hemos encontrado el elemento más pequeño, lo pasamos a R\$() y en T\$() lo sustituimos por el carácter 255.

```

1000 REM *****
1010 REM *
1020 REM * <<< ORDENACION POR EL METODO DE BUSQUEDA DEL MN >>> *
1030 REM *
1040 REM * VARIABLES QUE HAY QUE PASARLE AL PROGRAMA *
1050 REM * ----- *
1060 REM *
1070 REM * NU = NUMERO DE ELEMENTOS A ORDENAR *
1080 REM * T$( ) = TABLA CON LOS ELEMENTOS A ORDENAR *
1090 REM *
1100 REM * VARIABLES INTERNAS USADAS POR LA RUTINA *
1110 REM * ----- *
1120 REM *
1130 REM * A$ = VARIABLE DE INTERCAMBIO *
1140 REM * I = VARIABLE INDICE DEL PRIMER BUCLE *
1150 REM * A = VARIABLE AUXILIAR *
1160 REM * B = VARIABLE AUXILIAR *
1170 REM * C = VARIABLE AUXILIAR *
1180 REM * R$( ) = TABLA AUXILIAR *
1190 REM *
1200 REM * LA SUBROUTINA NOS RETORNA CON LOS DATOS *
1210 REM * ORDENADOS EN T$( ). *
1220 REM *
1230 REM *****
1240 REM
1250 DIM R$(NU)
1260 LET A=1
1270 LET M$=CHR$(255)
1280 LET B=1
1290 LET C=1
1300 IF C>NU-1 THEN GOTO 1350
1310 IF T$(B)<T$(C+1) THEN LET C=C+1:GOTO 1300
1320 LET B=C+1
1330 LET C=C+1

```

TRUCOS Y RUTINAS BASICAS

```
1340 GOTO 1300
1350 LET R$(A)=T$(B)
1360 LET T$(B)=M$
1370 LET A=A+1
1380 IF A<=NU THEN GOTO 1280
1390 FOR I=1 TO NU
1400     LET T$(I)=R$(I)
1410 NEXT I
1420 RETURN
```

Como siempre, la rutina puede funcionar en todos los ordenadores, pero para que funcione perfectamente en el SPECTRUM hay que variar el línea 1250 por:

1250 DIM R\$(NU,100)

Como ya dijimos un poco más arriba, el número 100 puede, y debe, variarse de acuerdo con la longitud de los elementos de la tabla.

La explicación del programa línea a línea es la siguiente:

Línea 1250. Dimensionamos la matriz R\$() que será la encargada de almacenar la tabla T\$() una vez que ésta esté ordenada.

Línea 1260. Hacemos que la variable A tenga valor uno. Esta variable será la encargada de apuntar a la posición dentro de R\$() donde tendremos que poner el siguiente elemento de T\$() que encontremos.

Línea 1270. Asignamos a la variable M\$ el carácter 255 y que utilizaremos para borrar de T\$() los elementos que ya han sido ordenados y almacenados en R\$().

Línea 1280. Hacemos que la variable numérica B tenga el valor 1. Esta variable será la que nos diga, en un momento cualquiera de la ejecución del programa, qué ficha es la que tiene un valor más pequeño de las que hemos visto hasta el momento.

Línea 1290. Asignamos el valor uno (1) a la variable C. Esta será la encargada de recorrerse la tabla T\$() de uno en uno hasta el final.

Línea 1300. Preguntamos si C es mayor que el número de elementos a ordenar menos uno (NU-1). En la primera vuelta del programa, esta línea no nos dice nada, pero será la encargada de controlar que ya hemos visto todos los elementos de T\$() y que, por tanto, ya tenemos almacenado en B la posición del más pequeño de ellos. Si resulta que ya nos hemos mirado toda la tabla, nos vamos a la línea 1350 para colocar el elemento más pequeño donde le corresponda.

Línea 1310. Preguntamos si el elemento que tenemos almacenado como el menor de

todos hasta el momento es menor que el que apunta C+1. En caso afirmativo, aumentamos en uno el valor de C para mirar el siguiente elemento y nos vamos a la línea 1300.

Línea 1320. Si resulta que el elemento al cual apuntamos con la C+1 es menor que el que está siendo apuntado por B, entonces hacemos que B tome la posición de dicho elemento.

Línea 1330. Hacemos que C se incremente en uno.

Línea 1340. Y nos volvemos a la línea 1300. Este grupo de líneas (desde la 1300 hasta la 1340) se va repitiendo hasta que hemos visto toda la lista y hemos encontrado cuál es el elemento más pequeño de todos. En ese momento nos vamos a la línea 1350.

Línea 1350. Colocamos el menor elemento en la tabla R\$() en la posición A.

Línea 1360. Para que dicho elemento no vuelva a salir, hacemos que sea igual al carácter 255 (M\$). Por ello hacemos la asignación: LET T\$(B)=M\$. Con ello conseguimos borrar este elemento.

Línea 1370. Aumentamos en uno el valor de A. Esto sirve para que cuando encontremos el siguiente elemento más pequeño de la lista, lo podamos colocar a continuación del anterior dentro de R\$().

Línea 1380. Si resulta que A es menor o igual que el número de elementos a ordenar, nos volvemos a la línea 1280 para hacer otra pasada a la lista en busca del siguiente elemento más pequeño. En esta línea y en la siguiente volvemos a poner a uno (1) el valor de las variables B y C.

Línea 1390. Si A es mayor que NU (número de elementos a ordenar) entonces significa que hemos terminado con la ordenación y por tanto tenemos que pasar el valor de la tabla R\$() a T\$(). En esta línea comenzamos un bucle para que haga esto mismo.

Línea 1400. Vamos pasando uno a uno los elementos de R\$() a T\$().

Línea 1410. Terminamos el bucle.

Línea 1420. Y devolvemos el control al

programa general y que llamó a esta rutina. Ya tenemos la lista ordenada totalmente en T\$().

El siguiente programa (programa 3) utiliza el método llamado SHELL METZNER para ordenar una serie de elementos. Este método

está especialmente indicado para aquellas listas que sean muy grandes y que no estén ordenadas de ninguna manera. Es muy veloz y utiliza un algoritmo matemático para ganar velocidad y para reducir el número de comprobaciones necesarias para ordenar una lista.

```

1000 REM *****
1010 REM *
1020 REM * <<< ORDENACION POR EL METODO DE SHELL METZNER >>> *
1030 REM *
1040 REM * VARIABLES QUE HAY QUE PASARLE AL PROGRAMA
1050 REM * -----
1060 REM *
1070 REM * NU = NUMERO DE ELEMENTOS A ORDENAR
1080 REM * T$( ) = TABLA CON LOS ELEMENTOS A ORDENAR
1090 REM *
1100 REM * VARIABLES INTERNAS USADAS POR LA RUTINA
1110 REM * -----
1120 REM *
1130 REM * A$ = VARIABLE DE INTERCAMBIO
1140 REM * I = VARIABLE INDICE DEL PRIMER BUCLE
1150 REM * A = VARIABLE AUXILIAR
1160 REM * B = VARIABLE AUXILIAR
1170 REM * C = VARIABLE AUXILIAR
1180 REM * D = VARIABLE AUXILIAR
1190 REM * E = VARIABLE AUXILIAR
1200 REM * N = VARIABLE AUXILIAR
1210 REM *
1220 REM * LA SUBROUTINA NOS RETORNA CON LOS DATOS
1230 REM * ORDENADOS EN T$( ).
1240 REM *
1250 REM *****
1260 REM
1270 FOR I=1 TO 16
1280 LET A=2^I
1290 IF A>NU THEN LET N=INT((2^I-1)/2):GOTO 1310
1300 NEXT I
1310 LET B=NU-N
1320 LET C=1
1330 LET D=C
1340 LET E=D+N
1350 IF T$(D)<=T$(E) THEN GOTO 1410
1360 LET A$=T$(D)
1370 LET T$(D)=T$(E)
1380 LET T$(E)=A$
1390 LET D=D-N
1400 IF D>=1 THEN GOTO 1340
1410 LET C=C+1
1420 IF C<=B THEN GOTO 1330
1430 LET N=INT(N/2)
1440 IF N<>0 THEN GOTO 1310
1450 RETURN

```

El algoritmo consiste en realizar las comparaciones entre elementos que se encuentran separados en la lista por una distancia constante. Esta distancia media, medida en elementos nos viene dada por la fórmula:

$$L = \text{INT}((2^N - 1) / 2)$$

donde N es un número tal que 2^N es superior al número de elementos que queremos ordenar, pero $2^{(N-1)}$ es menor que el número de elementos. Este número (N) es único y lo sacamos haciendo comprobaciones hasta que nos encontremos un N que al convertirlo en potencia de 2 nos dé un número mayor que el número de fichas a ordenar.

TRUCOS Y RUTINAS BASICAS

Por ejemplo, si queremos ordenar una lista con 30 elementos como:

$$2^5 = 32 > 30$$

entonces hacemos que la distancia media L sea igual a 16. Según esto, a la hora de comparar unos elementos con otros, compararemos el número 1 con el 17, el número 2 con el 18, el número 3 con el 19 y así sucesivamente. Cuando sea necesario intercambiaremos los dos elementos que estamos comparando. Una vez que hemos llegado al final y que hemos comparado todas las fichas de esta manera, reducimos el valor de L a la mitad (8 en nuestro ejemplo) y volvemos a repetir la misma operación. Esto se va repitiendo hasta que la distancia (L) es nula. En ese instante tendremos la tabla totalmente ordenada.

El programa puede funcionar en cualquier ordenador sin realizar ningún cambio.

La explicación línea a línea es la siguiente:

Línea 1270. Comenzamos el bucle entre 1 y 16 dentro del cual vamos a hallar la longitud media de la que hemos estado hablando.

Línea 1280. Hacemos que A sea igual a 2 elevado al valor de la variable índice del bucle.

Línea 1290. Si resulta que A es mayor o igual que el número de elementos a ordenar entonces es que acabamos de encontrar el exponente que buscábamos y hacemos que N sea la longitud media que necesitábamos. Como ya hemos encontrado dicha longitud, nos salimos del bucle y nos vamos a la línea 1310 para comenzar la ordenación.

Línea 1300. Aquí termina el bucle. Este bucle sólo está comprendido entre 1 y 16 porque es muy raro que un miniordenador (u ordenador personal) pueda tener una tabla con más de 2^{16} elementos (65535). En el caso de que su ordenador le permita tener más y que realmente necesite ordenar más de 65535 elementos, puede variar el intervalo del bucle entre 1 y el número que usted crea conveniente.

Línea 1310. Asignamos a la variable B el valor de $NU-N$. Esta variable será la encargada de almacenar el número de comparaciones que hay que hacer en cada pasada.

Línea 1320. Asignamos a la variable C el valor uno (1). Esta variable se encarga de contar el número de comparaciones que hemos hecho hasta el momento.

Línea 1330. Asignamos a la variable numérica D el valor que tenga C en ese momen-

to. Al principio del programa tendrá valor uno, pero según éste se vaya ejecutando, irá variando de valor. Esta variable nos sirve para saber cuál es el primer elemento que tenemos que comparar.

Línea 1340. Asignamos a la variable numérica E el valor de $D+N$. Esta variable nos indica el otro elemento de la lista que tenemos que comparar. Estas dos últimas líneas nos dicen qué dos elementos van a ser comparados a continuación. ER primer elemento está al principio de la lista y va aumentando con el tiempo, y el segundo elemento está N posiciones más adelante en la lista.

Línea 1350. Comparamos los dos elementos en cuestión. Si el que está más al principio de la lista es menor o igual que el que está más adelante, entonces no hay que invertir la posición de estos dos elementos. En ese caso nos vamos a la línea 1410 para continuar con el elemento siguiente.

Línea 1360. Si el elemento más cercano al principio de la lista es mayor que el que está más adelante, como tenemos que intercambiar las posiciones de ambos, hacemos que la variable alfanumérica auxiliar $A\$$ tome el valor del primero.

Línea 1370. Colocamos el que está más adelante en la lista en la posición del que estaba al principio de ésta.

Línea 1380. Y colocamos el que estaba al principio al final de la lista usando la variable $A\$$.

Línea 1390. A continuación le quitamos a la variable D el valor de la variable N .

Línea 1400. Preguntamos si D es mayor o igual que 1. En caso afirmativo nos vamos a la línea 1340 para continuar con las comprobaciones.

Línea 1410. En caso negativo, aumentamos en uno el contador de pasos C para comprobar los dos siguientes elementos de la lista.

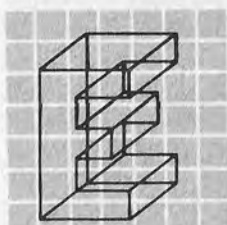
Línea 1420. Si no hemos hecho todavía todas las comprobaciones que tenemos que hacer nos volvemos a la línea 1330.

Línea 1430. En caso contrario dividimos por dos el valor de N para empezar a ordenar desde el nivel inferior.

Línea 1440. Si N es distinto de cero (0) significa que todavía no hemos terminado la ordenación con lo que nos vamos a la línea 1310 para continuar la misma.

Línea 1450. Si N era cero entonces la tabla está ordenada y devolvemos el control al programa principal.

■ Sistema de medidas de audiofrecuencia



N varios de los proyectos presentados se han propuestos circuitos que realizan funciones de filtrado de señales en el rango de baja frecuencia. Con los medios de que normalmente se dispone en el taller de hardware

normal no es corriente que se encuentre un osciloscopio, para verificar que el amplificador de filtro cumple las condiciones de diseño y que deja pasar exactamente la gama de frecuencias deseada. Vamos a presentar un circuito, que con algunas novedades sobre lo que ya se ha descrito, permitirá el análisis y control de cualquier sistema de audiofrecuencia, como podría ser el equipo de alta fidelidad. Podremos ver el efecto de los controles de tono o de ecualización gráfica.

El análisis de un circuito en baja frecuencia se realiza, generalmente, sometiéndole a una entrada senoidal de frecuencia conocida y analizando la respuesta. La salida que genera el sistema será también senoidal, de la misma frecuencia que la señal de entrada

pero de mayor amplitud, si es un amplificador, o menor, si es un atenuador. Respecto a la forma de la onda pueden aparecer deformaciones, que se denominan distorsión y retraso o adelanto de la fase.

Mediante el osciloscopio pueden realizarse medidas muy precisas de la ganancia y las deformaciones de la señal, de forma muy sencilla, pues la calibración en amplitud y tiempo dan de forma cuantificada el valor de las magnitudes de interés. Solamente la medida de distorsión requiere equipo especial, para poder apreciar el contenido de armónicos de la señal de salida o la comparación con una señal senoidal patrón que, restada de la señal a medir, indique la suma de las componentes introducidas por la no linealidad del amplificador.

Utilizando un ordenador es posible realizar de forma automática la medida de los principales parámetros que identifican el funcionamiento del circuito, dotándole de las interfaces apropiadas de conexión al equipo a medir.

Para el análisis de un equipo estereofónico se realizarán los ensayos de forma independiente en cada canal.

El conjunto de elementos necesarios para montar el sistema puede describirse mediante el esquema de bloques de la figura:

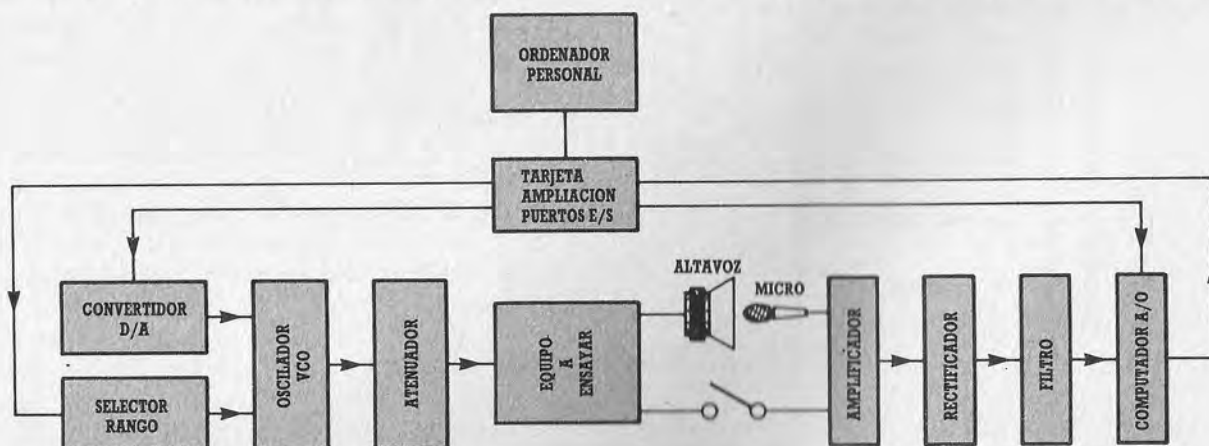


Fig. 1. Sistema de medidas de audiofrecuencia.

EL TALLER DEL HARDWARE

Las funciones realizadas por cada uno de los bloques son:

— Oscilador controlado. Genera la señal de referencia que se introducirá en el equipo a ensayar. Debe ser de características estables y repetibles. También debe dar una señal con muy bajo contenido en armónicos, para que el análisis sea adecuado. Se realiza generalmente con un oscilador controlado por tensión, la cual se genera mediante un conversor analógico/digital. Puede también controlarse manualmente, con objeto de economizar la conversión D/A, pero entonces la medida no podrá hacerse de forma totalmente automática.

Para cubrir un amplio margen de frecuencias, se conmuta un elemento del circuito en escalones de un factor conocido, por ejemplo, 10.

— Atenuador. Sirve para seleccionar el margen de valores de la señal de ensayo. Puede hacerse de forma automática, mediante conmutadores analógicos o manualmente con una resistencia variable. Como las medidas se realizan generalmente como variaciones sobre un valor de referencia, no resulta tan importante que se controle de forma automática.

— Equipo a probar. Puede ser cualquier sistema electrónico de baja frecuencia que disponga de una entrada y una salida.

— Micrófono. Una de las entradas posibles de análisis del equipo puede ser la señal acústica generada por los altavoces.

— Amplificador. Para adaptar a un nivel grande, para que los errores de medida tengan poca influencia, se amplifica la señal en diferentes escalones.

— Rectificador. La señal a medir se rectifica al valor de pico, mediante rectificador en doble onda.

— Filtro. La señal rectificada se filtra para quedarnos con la componente continua, correspondiente a la amplitud de la señal.

— Conversor Analógico/Digital. Para que el ordenador pueda utilizar el valor de la señal, se convierte a digital. Utilizaremos el mismo circuito empleado en un tomo anterior, que permite la entrada de varias señales analógicas.

— Interfaz. La entrada al ordenador se realiza a través de un puerto, direccionado de manera similar a los de proyectos anteriores.

— Ordenador personal. Puede ser cualquiera de los utilizados en los tomos anteriores, pues utilizando la tarjeta de ampliación de puertos de entrada/salida el circuito y la programación resultan los mismos para todas las máquinas, con las pequeñas excepciones mencionadas en cada caso.

A continuación se describen cada uno de los circuitos por separado, mostrándose al final un programa ejemplo, que puede servir de base para desarrollar cada uno el sistema de análisis ajustado a sus necesidades.

■ Oscilador controlado

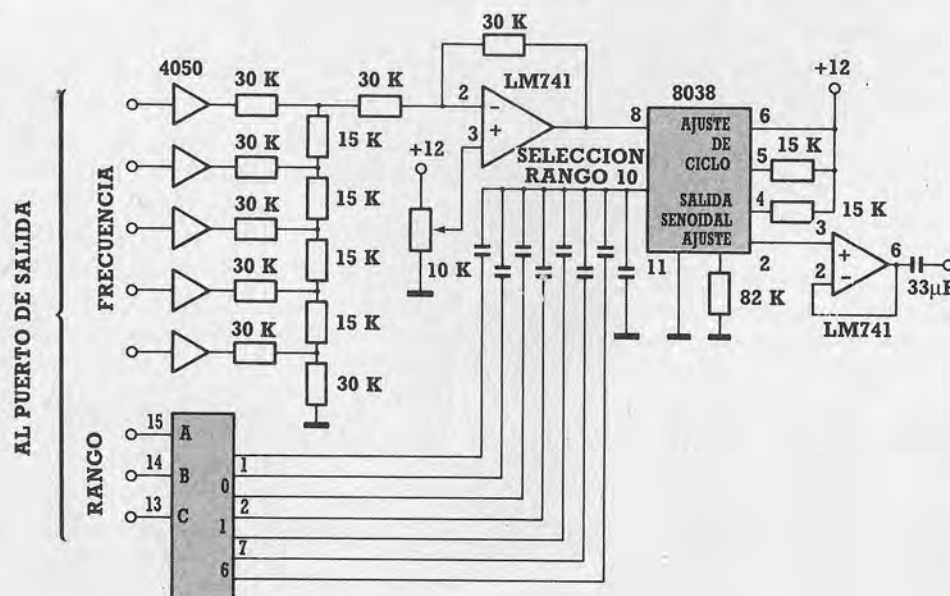


Fig. 2. Oscilador controlado.

Es el bloque fundamental del sistema. Se realiza mediante un conversor digital/análogo que podemos montar según se indica en la figura utilizando puertas y una red de resistencias del tipo R-2R. El número de escalones necesarios no es muy elevado, por lo que podemos realizar variaciones según las aplicaciones. En el circuito mostrado se pueden conseguir 32 diferentes niveles para cada valor de condensador. Como alternativa al nuevo circuito, podríamos utilizar el conversor mostrado en un tomo anterior.

La señal analógica generada por el conversor y amplificada por el circuito alimenta la entrada de control del oscilador. Mediante una resistencia variable a la entrada del circuito amplificador-sumador, se puede establecer el nivel de cero del oscilador.

El circuito oscilador es el ICL8038, que posee una gran variedad de formas de operación. Utilizado en la forma que se indica permite la obtención de una señal senoidal de frecuencia proporcional a la tensión analógica de entrada multiplicada por el valor del condensador que se conecte a la entrada 10.

Para poder variar el margen de oscilación se conectan varios condensadores en paralelo, pero con una de sus terminales conectada a un circuito integrado selector de tipo 7445. Las puertas de este circuito poseen a la salida un transistor en colector abierto. Si el transistor está sin alimentar, es decir, en corte, se comporta, para baja señal, como un circuito abierto de muy elevada impedancia. Si el transistor se encuentra alimentado se comporta como un cortocircuito. El montaje, por tanto, permite la conexión por programa de un condensador seleccionable a la entrada del oscilador. Los valores de los condensadores deberían hacer que pudiéramos generar frecuencias en el rango de 20 a 20.000 Hz. en saltos de décadas. Al no poderse cubrir una década con variaciones de tensión es necesario utilizar rangos más pequeños. Por supuesto que cualquier serie de valores permitirá realizar ensayos, pero si se puede explorar de forma regular, el análisis será más fácil. En cualquier caso, como los condensadores serán de tolerancia del 5-10% será necesario realizar una calibración para cada rango y el ajuste con condensadores en paralelo hasta conseguir que en cada posición la frecuencia generada guarde una relación constante.

Los rangos recomendables son los múltiplos de 1, 2 y 5. También con saltos en múltiplos de 3 podremos cubrir el espectro necesario.

tiplos de 3 podremos cubrir el espectro necesario.

La señal de control del oscilador debe estar comprendida entre 8 y 12 voltios, por lo que la resistencia variable conectada a la entrada positiva del amplificador permite ajustar el nivel apropiado para la frecuencia más baja en cada rango. El margen de variación es de 4 a 1.

Los amplificadores operacionales se alimentarán con tensiones de +12 y -12 voltios. Los circuitos lógicos con +5 voltios.

■ Atenuador

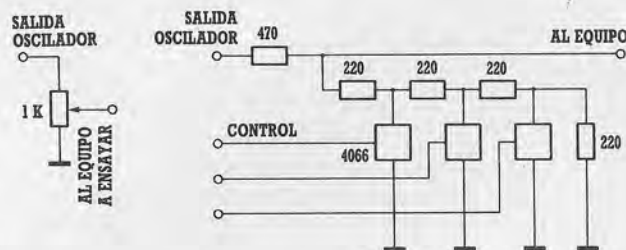


Fig. 3. Atenuador.

El sistema conviene que sea lo más general posible, para que podamos realizar medidas en todos los circuitos que sean de las características indicadas. Para ello la señal de alimentación al circuito deberá poder variarse dentro de un amplio rango. En la figura se muestra un circuito de control manual, que es simplemente una resistencia variable conectada a la salida del oscilador y que calibraremos con un pequeño dial.

Se muestra también una posible forma de realizar un atenuador controlado por programa, para un sistema totalmente automático. Utiliza conmutadores analógicos, de tipo 4066, que varían el valor de la resistencia conectada y, por tanto, el de la señal de salida del atenuador.

■ Equipo a probar



Fig. 4. Conexión del equipo a probar.

EL TALLER DEL HARDWARE

Para poder probar diferentes equipos montaremos un panel con los conectores de la señal de salida y de la entrada de micrófono o auxiliar. De particular importancia es la conexión de la masa. En un amplificador típico las señales de entrada y salida son de valores muy diferentes. La señal de entrada puede ser de unos pocos milivoltios, mientras que la de salida puede ser de decenas de voltios. La corriente que puede circular por la carga o el altavoz de salida puede ser de algunos amperios, en cuyo caso, la más mínima resistencia en los circuitos comunes a la entrada y la salida puede producir realimentaciones, que pueden resultar desastrosas para el equipo. Para estos casos, conviene realizar las medidas con circuitos aislados. Si la señal de salida se toma de los altavoces mediante un micrófono, el problema desaparece, aunque entonces hay que considerar a dicho micrófono como parte integrante del sistema.

El conector más conveniente para la conexión del equipo a ensayar es el de tipo RCA, empleado en muchos equipos de audio para las entradas de fono y auxiliar. Es además muy económico.

■ Micrófono

Se utiliza como elemento captador de la señal amplificada por el sistema, cuando el equipo a ensayar es un amplificador de audio que alimenta a un conjunto de altavoces. Conviene que sea de características muy lineales y de ancho de banda mayor que el del sistema. Los micros actuales de tipo "electret" son económicos y de características idóneas para estos ensayos. Se colocará en las proximidades del altavoz, para que la influencia del recinto donde realicemos el ensayo sea la mínima.

La otra entrada se usará para conexión de equipos a ensayar directamente con señal eléctrica.

■ Amplificador

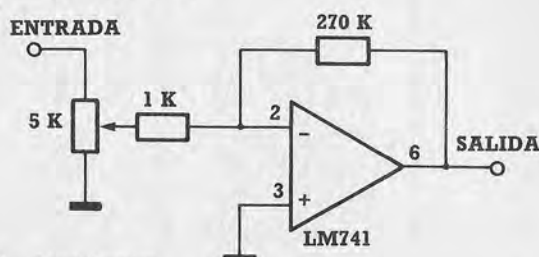


Fig. 5. Amplificador.

La señal suministrada por el micrófono será muy pequeña para poder analizarla directamente por el conversor analógico/digital. Es necesario amplificarla hasta niveles adecuados, superiores a 3 voltios. Se consigue la amplificación suficiente con una etapa con amplificador operacional. Conviene también disponer un atenuador variable, para ajustar el nivel de amplificación. Es importante, sin embargo, no modificar la posición del atenuador, una vez que se verifique que podemos trabajar dentro del rango de tensiones adecuado. Durante un ensayo completo las posiciones de los atenuadores deberán estar en posición fija.

■ Rectificador

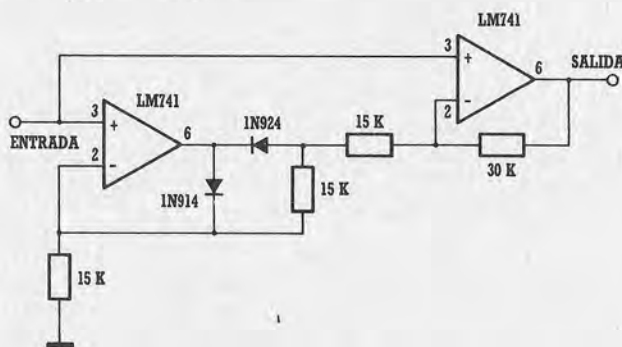


Fig. 6. Circuito rectificador.

La señal amplificada se filtra en doble onda, mediante un par de amplificadores operacionales montados en rectificador. El primer amplificador rectifica la señal en media onda, mientras que el segundo suma la señal de entrada a la señal rectificada, dando a la salida el valor absoluto de la entrada.

■ Filtro

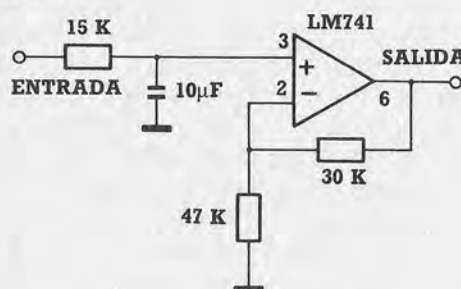
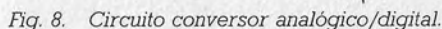


Fig. 7. Circuito filtro.

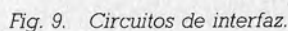
La señal rectificada se filtra con una etapa simple de resistencia condensador, que

■ Conversor analógico/digital



circuito mostrado se inicia por programa, al direccionar el canal a medir. La señal de estado indica cuándo la conversión ha terminado.

■ Interfaz



EL TALLER DEL HARDWARE

Los circuitos de interfaz son los mismos que los mostrados en la tarjeta de ampliación de puertos de entrada/salida. Se generan los pulsos de selección de los puertos mediante decodificadores, para conseguir las direcciones SELW0 y SELW1 para los puertos de salida y SELR0 y SELR1 para los de lectura.

Los puertos de salida contienen los bits de control y de selección de la frecuencia de acuerdo con la siguiente lista:

Puerto de salida 0:

- DSA0-DA4: Valor de la frecuencia. Salida al conversor D/A de control de la frecuencia.

- DSA5-DA7: Rango de frecuencias. Selección de condensador.

Puerto de salida 1:

- DSB0: Activación del conversor A/D.

Puerto de entrada 0:

- DEA0-DEA7: Entrada del conversor A/D.

Puerto de entrada 1:

- DEB0: Bit de estado del conversor A/D.

■ Consideraciones prácticas

Todos los amplificadores operacionales pueden ser del tipo LM741, pues las frecuencias empleadas son bajas. Es recomendable el empleo de las versiones integradas que incluyen 4 circuitos iguales en un solo chip. Si se desea extender el rango de frecuencias a analizar, será necesario sustituir el 741 por uno de mayor frecuencia de corte.

Las alimentaciones empleadas son +5, +12 y -12 voltios. Deberán ser estabilizadas y desacopladas con condensadores de 1 micro y 0.1 micro en las proximidades de los circuitos integrados.

El montaje puede realizarse en circuito de tiras, teniendo especial cuidado en la par-

te de señal de bajo nivel y conexión al micrófono, que deberían ir apantalladas.

Si se utiliza conectado a la tarjeta de ampliación de puertos de entrada/salida, el panel de conexiones puede fijarse directamente sobre la tarjeta. Si se monta en tarjeta a introducir en el ordenador, el panel de conexiones se montará en el exterior, uniendo las señales de bajo nivel mediante cable apantallado.

El ajuste del circuito requiere el ensayo paso a paso de un circuito conocido, que puede ser simplemente un filtro pasivo con resistencia condensador. Ejecutando el programa ejemplo deberemos obtener una curva de características planas para la conexión directa y otra de ganancia decreciente con la frecuencia, para un filtro paso bajo. Podemos modificar el programa para ejercitar la producción de cada frecuencia y la medida en condiciones estáticas de los valores en todos los puntos del circuito. Solamente cuando hayamos verificado el barrido completo, podremos analizar un circuito real.

■ Programa de ejemplo

Para la utilización de la tarjeta de interfaz es necesario programar cada operación y almacenar los resultados obtenidos para su presentación. Mediante un sencillo menú se selecciona la función a ejecutar. En cada función se activan los puertos necesarios para la producción de la señal senoidal de prueba, dentro del rango de frecuencias del ensayo. En el ejemplo se muestra el caso de análisis en el rango completo de frecuencias desde 20 a 20.000 Hz, en tres décadas, con saltos correspondientes a los múltiplos de 1, 2 y 5. Se supone que se han ajustado los condensadores para que se verifiquen los valores de las frecuencias necesarias.

La lectura del valor analógico en cada medida se efectúa un número repetido de veces, con objeto de filtrar el ruido. El número de repeticiones puede variarse mediante la variable NMAX.

```
10 REM * * * * *
20 REM PROGRAMA DE ANALISIS DE CIRCUITOS AUDIOFRECUENCIA
30 REM * * * * *
40 REM UTILIZA LA TARJETA DE AMPLIACION DE PUERTOS
50 REM EN EL IBM PC
60 REM * * * * *
70 DEFINT A-Z
100 REM DEFINICION DE PUERTOS
```

```

101 DIM G(500)
107 PDATA=&H201: REM DATOS - SELRO
108 PCONTROL=&H201: REM RANGO Y FRECUENCIA - SELWO
109 PSELECCION=&H203: REM NUMERO DE CANAL - SELW1
110 PESTADO=&H203: REM ESTADO DEL CONVERSOR - SELR1
120 REPOSO=0
128 KEY OFF
129 CLS
130 PRINT " PROGRAMA DE ANALISIS DE CIRCUITOS AUDIOFRECUENCIA "
140 PRINT
143 PRINT " VERIFIQUE LAS CONEXIONES DE ENTRADA Y SALIDA "
147 PRINT
150 PRINT " SELECCIONE OPCION:"
160 PRINT
170 PRINT " 1.-EFECTUAR MEDIDAS"
190 PRINT
200 PRINT " 2.-PRESENTAR RESULTADOS"
210 PRINT
220 PRINT " 3.-TERMINAR"
260 PRINT
270 INPUT "OPCION=":OPCION
280 IF OPCION<1 OR OPCION>3 THEN BEEP: GOTO 129
290 ON OPCION GOSUB 1000,2000,3000
295 IF OPCION=3 THEN END
300 GOTO 129
990 REM * * * * EFECTUAR MEDIDAS * * * *
1000 CLS:
1010 PRINT " MEDIDAS DE GANANCIA"
1020 PRINT
1170 REM ESTABLECE RANGO Y FRECUENCIA
1200 NMAX=8
1210 FOR I=0 TO 7: REM RANGO
1220   FOR J=0 TO 31: REM FRECUENCIA
1230     OUT PCONTROL,(I+J)
1235     A=0
1240     FOR K=1 TO NMAX: REM MEDIDA
1250       OUT PSELECCION,0: REM ACTIVA CONVERSION
1260       A=A+INP(PDATA)
1270     NEXT K
1280     G(I*32+J)=A/NMAX
1290   NEXT J
1300 NEXT I
1310 PRINT "FIN DE TOMA DE DATOS"
1500 RETURN
1990 REM * * * * PRESENTA RESULTADOS * * * *
2000 CLS
2010 PRINT "VECTOR DE GANANCIAS"
2020 PRINT
2030 FOR I=1 TO 32*8
2240   PRINT "I=";I;" G=";G(I)
2300 NEXT I
2310 RETURN
2990 REM * * * * FUNCION DE PUESTA EN REPOSO * * * *
3000 REM DEJA LA TARJETA EN REPOSO
3010 OUT PCONTROL,REPOSO
3100 RETURN

```

Conclusiones

Se ha mostrado un circuito, de cierta complejidad, pero que utiliza módulos que ya hemos empleado en proyectos anteriores. Tiene la particularidad de que nos permite realizar medidas en sistemas reales de forma enteramente automática, para obtener la curva característica de ganancia en función de la frecuencia. Se suele denominar a esta curva diagrama de Bode, pues es el nombre del que realizó estudios de la influencia de dicha curva en la estabilidad de sistemas realimenta-

dos. La representación gráfica de dicha característica es importante para conocer las prestaciones de los circuitos y determinar de forma empírica los resultados de diseños teóricos, o de la calidad esperable en la reproducción de señales acústicas.

Como posibles variaciones a realizar en el circuito para mejorar sus características podríamos indicar:

— Control del oscilador con un convertidor D/A en circuito integrado, en lugar de realizarlo con puertas y red de resistencias. Con el circuito DAC0801 se podrían conseguir 256

valores de frecuencia, para cada rango de condensadores, con aumento en la resolución de la medida.

- Mayor número de condensadores, para aumentar el rango de frecuencias de exploración, hasta el límite del oscilador controlado por tensión.

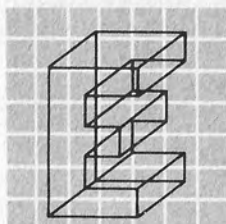
- Selección de rectificación de la señal al valor medio, de pico o eficaz, mediante los circuitos adecuados, para tener mayor conocimiento de las posibles deformaciones de la señal.

- Conversor A/D más rápido, para poder realizar muestreo de la señal y poder analizar la forma de onda producida. Con la resolución adecuada podríamos hacer digitalmente el análisis del contenido en armónicos de la señal. En un próximo tomo indicaremos algunas de las posibilidades de tratamiento digital de señales, mediante nuestro ordenador.

- Conmutación a dos canales, para poder realizar ensayos en equipos estéreo o en dos equipos iguales a la vez, permitiendo el ajuste o calibración "en tiempo real".

NATURALEZA Y TECNOLOGIA

Asociación de resistencias



STE programa calcula la resistencia equivalente de un circuito formado por resistencias. Complementariamente calcula la intensidad que circula por cada una de las resistencias.

Antes de empezar a introducir datos en el programa es necesario configurar el circuito por bloques. ¿Qué es un bloque? Un bloque lo constituye cualquier asociación elemental de resistencias, es decir, cualquier grupo de resistencias conectadas únicamente en serie o únicamente en paralelo.

lo. Después de introducir las resistencias de cada bloque, el programa preguntará cómo están relacionados los distintos bloques (en serie o en paralelo).

Una vez que se impriman por pantalla los resultados tanto de la resistencia equivalente del circuito como de la resistencia equivalente de cada bloque, el programa preguntará cuál es el valor en voltios del generador de corriente continua que está suministrando tensión al circuito. Este dato es necesario para conocer qué intensidad circula por cada una de las partes de nuestro circuito. Finalmente el programa imprimirá el valor de la intensidad en cada una de las resistencias.

El siguiente ejemplo ilustra todo lo anterior:

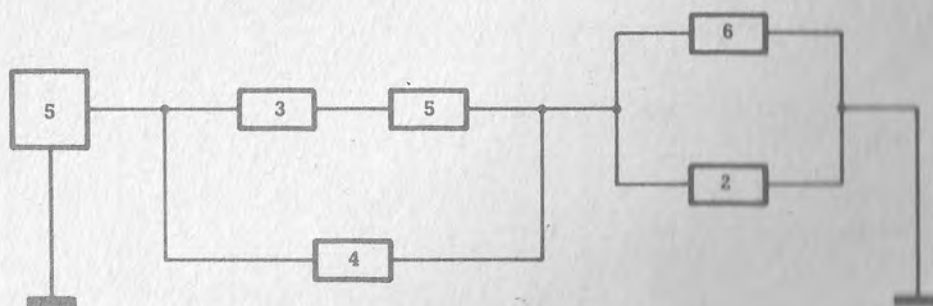


Fig. 1.

```

DE CUANTOS BLOQUES CONSTA EL CIRCUITO? 3
BLOQUE 1
ASOCIACION (S)ERIE 0 (P)ARALELO ? S
DE CUANTAS RESISTENCIAS CONSTA ? 2
INTRODUZCA EL VALOR EN OHMIOS
? 3
? 5
BLOQUE 2
ASOCIACION (S)ERIE 0 (P)ARALELO ? S
DE CUANTAS RESISTENCIAS CONSTA ? 1
INTRODUZCA EL VALOR EN OHMIOS
? 4
BLOQUE 3
ASOCIACION (S)ERIE 0 (P)ARALELO ? P
DE CUANTAS RESISTENCIAS CONSTA ? 2
INTRODUZCA EL VALOR EN OHMIOS
? 6
? 2
RELACION ENTRE LOS DISTINTOS BLOQUES
INTRODUCIR RELACION ENTRE BLOQUE ANTERIOR Y EL BLOQUE 2 'S' 0 'P' ? P
INTRODUCIR RELACION ENTRE BLOQUE ANTERIOR Y EL BLOQUE 3 'S' 0 'P' ? S
RESISTENCIA POR BLOQUES
BLOQUE 1 = 8 OHM.
BLOQUE 2 = 4 OHM.
BLOQUE 3 = 1.5 OHM.
LA RESISTENCIA TOTAL ES 4.166667 OHM.
INTRODUZCA EL VALOR DEL GENERADOR DE CORRIENTE CONTINUA ? 5
LA INTENSIDAD TOTAL QUE CIRCULA POR EL CIRCUITO ES 1.2 AMP.
INTENSIDAD TOTAL DE CADA BLOQUE
BLOQUE 1 = .4 AMP.
BLOQUE 2 = .8 AMP.
BLOQUE 3 = 1.2 AMP.
INTENSIDAD POR RESISTENCIA
BLOQUE 1
R 1 = 3 OHM. I 1 = .4 AMP.
R 2 = 5 OHM. I 2 = .4 AMP.
BLOQUE 2
R 3 = 4 OHM. I 3 = .8 AMP.
BLOQUE 3
R 4 = 6 OHM. I 4 = .3 AMP.
R 5 = 2 OHM. I 5 = .9 AMP.

```

```

10 REM *****
20 REM * TITULO:ASOCIACION DE RESISTENCIAS *
30 REM * VERSION:1.0 *
40 REM * PARA IBM-PC, MSX, AMSTRAD *
50 REM * COMMODORE, SPECTRUM *
60 REM *****
100 REM *****
110 REM * INICIALIZACION *
120 REM *****
130 LET R=0
140 LET T=0
150 DIM R(50)
160 INPUT"DE CUANTOS BLOQUES CONSTA EL CIRCUITO";N
170 DIM B(N,2)
180 DIM B$(N)
190 DIM T(N)
200 DIM A(N)
300 REM *****
310 REM * DATOS DE LOS BLOQUES *
320 REM *****
330 FOR I=1 TO N
340 PRINT"BLOQUE ";I
350 INPUT"ASOCIACION (S)ERIE 0 (P)ARALELO ";A$
360 IF A$<>"S" AND A$<>"P" THEN PRINT"PULSAR 'S' 0 'P'":GOTO 350

```

```

370 IF A$="S" THEN GOSUB 1000
380 IF A$="P" THEN GOSUB 1210
390 NEXT I
400 REM *****
410 REM * RELACION ENTRE LOS BLOQUES *
420 REM *****
430 PRINT"RELACION ENTRE LOS DISTINTOS BLOQUES"
440 LET T=B(1,2)
450 LET T(1)=T
460 IF N=1 THEN GOTO 610
470 FOR I=2 TO N
480 PRINT"INTRODUCIR RELACION ENTRE BLOQUE ANTERIOR Y EL BLOQUE ";I;
  " 'S' O 'P' ";
490 INPUT A$
500 IF A$<>"S" AND A$<>"P" THEN PRINT"PULSAR 'S' O 'P'":GOTO 480
510 IF A$="S" THEN GOSUB 1530
520 IF A$="P" THEN GOSUB 1630
530 NEXT I
600 REM *****
610 REM * RESISTENCIA POR BLOQUES *
620 REM *****
630 PRINT"RESISTENCIA POR BLOQUES"
640 FOR I=1 TO N
650 PRINT"BLOQUE ";I;" = ";B(I,2);" OHM."
660 NEXT I
700 REM *****
710 REM * RESISTENCIA TOTAL *
720 REM *****
730 PRINT"LA RESISTENCIA TOTAL ES ";T;" OHM."
740 INPUT"INTRODUZCA EL VALOR DEL GENERADOR DE CORRIENTE CONTINUA ";V
750 PRINT"LA INTENSIDAD TOTAL QUE CIRCULA POR EL CIRCUITO ES ";V/T;" AMP."
800 REM *****
810 REM * INTENSIDAD POR BLOQUES *
820 REM *****
830 GOSUB 1710
840 PRINT"INTENSIDAD TOTAL DE CADA BLOQUE"
850 FOR I=1 TO N
860 PRINT"BLOQUE ";I;" = ";A(I);" AMP."
870 NEXT I
900 REM *****
910 REM * INTENSIDAD POR RESISTENCIA *
920 REM *****
930 PRINT"INTENSIDAD POR RESISTENCIA"
940 LET X=0
950 FOR I=1 TO N
960 IF B$(I)="S" THEN GOSUB 2200
970 IF B$(I)="P" THEN GOSUB 2100
980 NEXT I
990 END
1000 REM *****
1010 REM * BLOQUE SERIE *
1020 REM *****
1030 LET B$(I)="S"
1040 INPUT"DE CUANTAS RESISTENCIAS CONSTA ";B(I,1)
1050 GOSUB 1430
1100 REM *****
1110 REM * CALCULO DE LA RESISTENCIA BLOQUE SERIE *
1120 REM *****
1130 LET B(I,2)=0
1140 FOR M=1 TO B(I,1)
1150 LET B(I,2)=B(I,2)+R(R+1-M)
1160 NEXT M
1170 RETURN
1200 REM *****
1210 REM * BLOQUE PARALELO *
1220 REM *****
1230 LET B$(I)="P"
1240 INPUT"DE CUANTAS RESISTENCIAS CONSTA ";B(I,1)
1250 GOSUB 1430
1300 REM *****
1310 REM * CALCULO DE LA RESISTENCIA BLOQUE PARA. *
1320 REM *****
1330 LET B(I,2)=R(R)
1340 IF B(I,1)=1 THEN GOTO 1380
1350 FOR M=2 TO B(I,1)
1360 LET B(I,2)=B(I,2)+R(R+1-M)/(B(I,2)+R(R+1-M))
1370 NEXT M
1380 RETURN
1400 REM *****
1410 REM * DATOS DE LAS RESISTENCIAS *
1420 REM *****
1430 PRINT"INTRODUZCA EL VALOR EN OHMIOS"
1440 FOR K=1 TO B(I,1)

```

```

1450 LET R=R+1
1460 INPUT R(R)
1470 NEXT K
1480 RETURN
1500 REM *****
1510 REM * ASOCIACION BLOQUES SERIE *
1520 REM *****
1530 REM
1540 LET T$(I)="S"
1550 LET T=T+B(I,2)
1560 LET T(I)=T
1570 RETURN
1600 REM *****
1610 REM * ASOCIACION BLOQUES PARALELO *
1620 REM *****
1630 REM
1640 LET T$(I)="P"
1650 LET T=T*B(I,2)/(T+B(I,2))
1660 LET T(I)=T
1670 RETURN
1700 REM *****
1710 REM * CALCULO DE LA INTENSIDAD POR BLOQUES *
1720 REM *****
1730 LET A=V/T
1740 IF N=1 THEN GOTO 1790
1750 FOR I=N TO 2 STEP -1
1760 IF T$(I)="S" THEN GOSUB 1930
1770 IF T$(I)="P" THEN GOSUB 2030
1780 NEXT I
1790 LET A(1)=A
1800 RETURN
1900 REM *****
1910 REM * INTENSIDAD SERIE *
1920 REM *****
1930 LET A(I)=A
1940 RETURN
2000 REM *****
2010 REM * INTENSIDAD PARALELO *
2020 REM *****
2030 LET A(I)=A*T(I)/B(I,2)
2040 LET A=A*T(I)/T(I-1)
2050 RETURN
2100 REM *****
2110 REM * INTENSIDAD POR RESISTENCIA PARALELO *
2120 REM *****
2130 PRINT "BLOQUE ";I
2140 FOR J=X+1 TO X+B(I,1)
2150 PRINT "R";J;"=";"R(J);" OHM. I";J;"=";"A(I)*B(I,2)/R(J);" AMP."
2160 NEXT J
2170 LET X=X+B(I,1)
2180 RETURN
2200 REM *****
2210 REM * INTENSIDAD POR RESISTENCIA SERIE *
2220 REM *****
2230 PRINT "BLOQUE ";I
2240 FOR J=X+1 TO X+B(I,1)
2250 PRINT "R";J;"=";"R(J);" OHM. I";J;"=";"A(I);" AMP."
2260 NEXT J
2270 LET X=X+B(I,1)
2280 RETURN

```

MATEMATICAS

■ Cálculo de determinantes

Este programa calcula determinantes de orden menor o igual que cuatro. Para determinantes de orden mayor sería necesario programar procesos recursivos con las rutinas existentes ya en este programa.

Para los determinantes de orden dos, dada su sencillez, el programa directamente los calcula sin utilizar ninguna fórmula recursiva.

Con los demás determinantes se establece el proceso recursivo que consiste en ir dividiendo el determinante en adjuntos de orden tres, e ir multiplicándolos por su menor complementario.

```

10 REM *****
20 REM *      TITULO:CALCULO DE DETERMINANTES *
30 REM *      VERSION:2.0 *
40 REM *      PARA IBM-PC, MSX, AMSTRAD, *
50 REM *      COMMODORE, SPECTRUM *
60 REM *****
100 REM *****
110 REM * ORDEN DEL DETERMINANTE *
120 REM *****
130 INPUT"ORDEN DEL DETERMINANTE (SOLO HASTA CUATRO) ";O
140 IF O>4 OR O<2 THEN PRINT"SOLO HASTA ORDEN CUATRO":GOTO 130
150 DIM X(0,0),Y(0,0),A(0,0),Z(0)
200 REM *****
210 REM * VALORES DEL DETERMINANTE *
220 REM *****
230 PRINT"INTRODUZCA LOS VALORES DEL DETERMINANTE"
240 FOR F=1 TO O
250 FOR C=1 TO O
260 PRINT"FILE ";F;
270 PRINT"COLUMNA ";C;
280 INPUT" ";X(F,C)
290 NEXT C
300 NEXT F
310 IF O>2 THEN GOTO 450
400 REM *****
410 REM * DETERMINANTE DE ORDEN 2 *
420 REM *****
430 LET R=X(1,1)*X(2,2)-X(1,2)*X(2,1)
440 GOTO 1030
450 IF O>3 THEN GOTO 710
500 REM *****
510 REM * DETERMINANTE DE ORDEN 3 *
520 REM *****
530 FOR I=1 TO O
540 FOR J=1 TO O
550 LET A(I,J)=X(I,J)
560 NEXT J
570 NEXT I
580 GOSUB 1210
590 LET R=P
600 GOTO 1030
700 REM *****
710 REM * DETERMINANTE DE ORDEN 4 *
720 REM *****
730 FOR D=1 TO O
740 GOSUB 1310
750 FOR F=2 TO O
760 FOR C=1 TO O
770 LET Y(F-1,C)=Y(F,C)
780 NEXT C
790 NEXT F
800 FOR F=1 TO O-1
810 FOR C=1 TO O-1
820 IF D<>0 THEN Y(F,C)=Y(F,C+1)
830 NEXT C
840 NEXT F
850 GOSUB 1110
860 GOSUB 1210
870 LET Z(D)=P
880 NEXT D
890 FOR F=1 TO O
900 LET R=R+(-1)^(1+I)*Z(F)*X(1,F)
910 NEXT F
1000 REM *****
1010 REM * SALIDA DEL RESULTADO *
1020 REM *****
1030 PRINT"EL VALOR DEL DETERMINANTE ES ";R
1040 END
1100 REM *****
1110 REM * CALCULO DE ADJUNTOS DE ORDEN 3 *
1120 REM *****
1130 FOR I=1 TO O-1
1140 FOR J=1 TO O-1
1150 LET A(I,J)=Y(I,J)
1160 NEXT J
1170 NEXT I
1180 RETURN
1200 REM *****
1210 REM * CALCULO DEL DETERMINANTE DE ORDEN 3 *
1220 REM *****
1230 LET P=A(1,1)*(A(2,2)*A(3,3)-A(2,3)*A(3,2))
1240 LET P=P-A(1,2)*(A(2,1)*A(3,3)-A(2,3)*A(3,1))

```

```

1250 LET P=P+A(1,3)*(A(2,1)*A(3,2)-A(2,2)*A(3,1))
1260 RETURN
1300 REM *****
1310 REM * DETERMINANTE AUXILIAR 2 *
1320 REM *****
1330 FOR I=1 TO 0
1340 FOR C=1 TO 0
1350 LET Y(I,J)=X(I,J)
1360 NEXT J
1370 NEXT I
1380 RETURN

```

SOCIEDAD

■ Ciudades de España

Es curioso comprobar que, frecuentemente, nuestro conocimiento de la geografía de España se reduce a nuestro ámbito local, o a lo sumo, a las capitales de provincia más importantes.

El programa contiene una base de datos con tres de las ciudades más importantes de cada una de las provincias españolas.

Cada vez que se ejecute, el programa pedirá el nombre de una ciudad distinta de la capital, de la provincia que pregunte. Una vez

introducida la solución indicará si ésta es o no correcta e imprimirá las tres ciudades contenidas en su pequeña base de datos. Habrá diez preguntas, al final de las cuales el ordenador imprimirá el número de aciertos obtenidos.

El desarrollo del programa es bastante sencillo. Consta de una subrutina para leer los datos de la base de datos, contenidos en líneas 'DATA' del final del programa, y de un bloque principal que se encarga de elegir al azar una provincia y de comprobar que la respuesta sea correcta.

Esperamos que este programa te sirva para conocer un poco mejor la geografía de España.

```

10 REM *****
20 REM * TITULO:CIUDADES DE ESPAÑA *
30 REM * VERSION:1.0 *
40 REM * PARA IBM-PC, AMSTRAD, MSX *
60 REM *****
100 REM *****
110 REM * INTRODUCCION *
120 REM *****
130 CLS
140 PRINT"CON ESTE PROGRAMA SE PRETENDE DAR"
150 PRINT"UN REPASO A SUS CONOCIMIENTOS"
160 PRINT"DE GEOGRAFIA. CONCRETAMENTE SE LE"
170 PRINT"PREGUNTARA EL NOMBRE DE ALGUNA"
180 PRINT"CIUDAD DISTINTA DE LA CAPITAL DE"
190 PRINT"DIEZ PROVINCIAS ESPAÑOLAS"
200 REM *****
210 REM * RUTINA DE CARGA DE DATOS *
220 REM *****
230 DIM A$(50,4)
240 FOR I=1 TO 50
250 FOR J=1 TO 4
260 READ A$(I,J)
270 NEXT J
280 NEXT I
300 REM *****
310 REM * RUTINA GENERAL *
320 REM *****
330 LET P=0
340 LET S=0
350 FOR I=1 TO 10
360 LET A=INT(RND(1)*50)+1
370 PRINT"¿CUAL ES UNA CIUDAD DE ";A$(A,1);" ";
380 INPUT R$
390 FOR J=2 TO 4
400 IF R$=A$(A,J) THEN P=1
410 NEXT J
420 IF P=1 THEN PRINT"MUY BIEN":LET S=S+1
430 IF P=0 THEN PRINT"NO ES CORRECTO"
440 LET P=0
450 PRINT"LAS SOLUCIONES POSIBLES SON:"

```

```

460 FOR J=2 TO 4
470 PRINT A$(A,J)
480 NEXT J
490 NEXT I
500 PRINT "HAS ACERTADO ";S;" VECES"
1000 REM *****
1010 REM * DATOS *
1020 REM *****
1030 DATA LA CORUÑA,SANTIAGO,BETANZOS,FERROL
1040 DATA PONTEVEDRA,VIGO,TUY,CAMBADOS
1050 DATA LUGO,MONFORTE,BECERREA,SARRIA
1060 DATA ORENSE,GINZO,RIBADAVIA,VERIN
1070 DATA ASTURIAS,GIJON,MIERES,CANGAS
1080 DATA CANTABRIA,TORRELAVEGA,LAREDO,SAN VICENTE
1090 DATA VIZCAYA,DURANGO,GUERNICA,SESTAO
1100 DATA GUIPUZCOA,IRUN,TOLOSA,VERGARA
1110 DATA ALAVA,LAGUARDIA,AMURRIO,SALVATIERRA
1120 DATA NAVARRA,TAFALLA,ESTELLA,AOIZ
1130 DATA LA RIOJA,NAJERA,CALAHORRA,SANTO DOMINGO
1140 DATA HUESCA,BARBASTRO,JACA,SARIÑENA
1150 DATA ZARAGOZA,CARIÑENA,CALATAYUD,DAROCA
1160 DATA TERUEL,ALBARRACIN,ALCAÑIZ,CALAMUCHA
1170 DATA BARCELONA,MANRESA,TARRASA,SABADELL
1180 DATA GERONA,OLOT,FIGUERAS,SANTA COLOMA
1190 DATA LERIDA,VIELLA,BALAGUER,CERVERA
1200 DATA TARRAGONA,REUS,TORTOSA,AMPOSTA
1210 DATA VALENCIA,ALCIRA,JATIVA,SAGUNTO
1220 DATA CASTELLON,VINAROS,MORELLA,NULES
1230 DATA ALICANTE,ALCOY,ELCHE,ORIHUELA
1240 DATA MURCIA,CARTAGENA,LORCA,YECLA
1250 DATA ALMERIA,PURCHENA,BERJA,VERA
1260 DATA GRANADA,MOTRIL,GUADIX,BAZA
1270 DATA JAEN,ANDUJAR,LINARES,LA CAROLINA
1280 DATA CORDOBA,MONTILLA,LA RAMBLA,CABRA
1290 DATA MALAGA,ANTEQUERA,ARCHIDONA,MARBELLA
1300 DATA SEVILLA,UTRERA,CARDONA,OSUNA
1310 DATA CADIZ,JEREZ,ARCOS,SAN FERNANDO
1320 DATA HUELVA,AYAMONTE,MOGUER,VALVERDE
1330 DATA LEON,PONFERRADA,ASTORGA,LA BANEZA
1340 DATA ZAMORA,TORO,BENAVENTE,PUEBLA DE SANABRIA
1350 DATA SALAMANCA,PEÑARANDA,BEJAR,LEDESMA
1360 DATA AVILA,CEBREROS,EL BARCO,PIEDRAHITA
1370 DATA SEGOVIA,SEPULVEDA,CUELLAR,RIAZA
1380 DATA SORIA,BURGO DE OSMÁ,ALMAZAN,MEDINACELI
1390 DATA BURGOS,ARANDA,ROA,BRIVIESCA
1400 DATA PALENCIA,ASTUDILLO,SALDAÑA,CARRION
1410 DATA VALLADOLID,MEDINA,OLMEDO,PEÑAFIEL
1420 DATA MADRID,GETAFE,EL ESCORIAL,ARANJUEZ
1430 DATA GUADALAJARA,SIGÜENZA,SACEDON,MOLINA
1440 DATA CUENCA,HUETE,MOTILLA,BELMONTE
1450 DATA ALBACETE,LA RODA,ALCARAZ,ALMANSÁ
1460 DATA CIUDAD REAL,PUERTOLLANO,VALDEPEÑAS,MANZANARES
1470 DATA TOLEDO,QUINTANAR,TALAVERA,ILLESCHAS
1480 DATA CACERES,PLASENCIA,NAVALMORAL,ALCANTARA
1490 DATA BADAJOZ,ZAFRA,MERIDA,ALMENDRALEJO
1500 DATA BALEARES,MAHON,IBIZA,MANACOR
1510 DATA TENERIFE,LA LAGUNA,LA OROTAVA,GRANADILLA
1520 DATA LAS PALMAS,GUIA,TELDE,ARRECIFE

```

Modificaciones para otros ordenadores

SPECTRUM

LINEA 230 DIM A\$(50,4,15)
 TODOS LOS DATOS DE LAS LINEAS 1030 A
 LA 1520 DEBERAN IR ENTRE
 COMILLAS.

PARA LOS MAS JOVENES

■ Rebote

Con este programa se pretende simular el rebote de un móvil al chocar contra los

bordes de la pantalla. En el primer programa se simula el movimiento del carácter $\odot$ en un trozo de pantalla cuyas dimensiones están fijadas en las variables U, D, L y R. Se usan las variables X e Y para las coordenadas del móvil en la pantalla. El movimiento que describe el móvil es diagonal; las variables I y K contienen el valor de la dirección y el sentido de esa diagonal. Se incluye un bucle de retardo en las líneas 380 y 390 para evitar que el móvil se mueva demasiado de prisa; la duración del retardo es variable modificando el límite del bucle.

APRENDER CON EL ORDENADOR

En la segunda versión del programa se ha introducido otro móvil; por ello son necesarias más variables de posición y de diagonal que en la primera versión.

Por último, en la tercera versión se ha introducido una subrutina para simular el

choque de los dos móviles cuando llegan a la misma posición de la pantalla.

Para el ordenador Commodore se ha escrito una cuarta versión equivalente a la tercera antes descrita.

```

10 REM *****
20 REM *      TITULO:REBOTE      *
30 REM *      VERSION:1.0      *
40 REM *      PARA IBM-PC Y COMPATIBLES      *
50 REM *****
100 REM *****
110 REM * INICIALIZACION DE LAS VARIABLES *
120 REM *****
130 LET U=2
140 LET D=18
150 LET L=2
160 LET R=25
170 LET X=4
180 LET Y=4
190 LET I=1
200 LET J=1
210 CLS
300 REM *****
310 REM * RUTINA GENERAL *
320 REM *****
330 GOSUB 500
340 LET X=X+I
350 LET Y=Y+J
360 LOCATE Y,X
370 PRINT"*";
380 FOR A=1 TO 50
390 NEXT A
400 LOCATE Y,X
410 PRINT " ";
420 GOTO 310
500 REM *****
510 REM * RUTINA DE COMPROBACION DE LOS BORDES *
520 REM *****
530 IF X=R OR X=L THEN LET I=-I
540 IF Y=U OR Y=D THEN LET J=-J
550 RETURN

```

Modificaciones para otros ordenadores

AMSTRAD Y MSX

LINEA 360 LOCATE X,Y
LINEA 400 LOCATE X,Y

SPECTRUM

LINEA 370 PRINT AT X,Y,"*";
LINEA 410 PRINT AT X,Y," ";
DESAPARECEN LAS LINEAS 360 Y 400

```

10 REM *****
20 REM *      TITULO:REBOTE      *
30 REM *      VERSION:1.1      *
40 REM *      PARA IBM-PC Y COMPATIBLES      *
50 REM *****
100 REM *****
110 REM * INICIALIZACION DE LAS VARIABLES *
120 REM *****
130 LET U=2
140 LET D=18
150 LET L=2
160 LET R=25
170 LET X=4

```

```

180 LET M=10
190 LET Y=4
200 LET N=10
210 LET I=1
220 LET K=1
230 LET J=1
240 LET P=-1
300 REM *****
310 REM * RUTINA GENERAL *
320 REM *****
330 GOSUB 530
340 GOSUB 630
350 LET X=X+I
360 LET M=M+K
370 LET Y=Y+J
380 LET N=N+P
390 LOCATE Y,X
400 PRINT"*";
410 LOCATE N,M
420 PRINT"D";
430 FOR A=1 TO 50
440 NEXT A
450 LOCATE Y,X
460 PRINT " ";
470 LOCATE N,M
480 PRINT " ";
490 GOTO 310
500 REM *****
510 REM *RUTINA DE COMPROBACION DE BORDES 1 *
520 REM *****
530 IF X=R OR X=L THEN LET I=-I
540 IF Y=U OR Y=D THEN LET J=-J

```

```

550 RETURN
600 REM *****
610 REM *RUTINA DE COMPROBACION DE BORDES 2 *
620 REM *****
630 IF M=R OR M=L THEN LET K=-K
640 IF N=U OR N=D THEN LET P=-P
650 RETURN

```

Modificaciones para otros ordenadores

Estas modificaciones son comunes para las versiones 2.0 y 1.1

AMSTRAD Y MSX

LINEA 390 LOCATE X,Y
 LINEA 410 LOCATE M,N
 LINEA 450 LOCATE X,Y
 LINEA 470 LOCATE M,N

SPECTRUM

LINEA 400 PRINT AT X,Y,;,
 LINEA 420 PRINT AT M,N;"0";
 LINEA 460 PRINT AT X,Y," ";
 LINEA 480 PRINT AT M,N," ";
 DESAPARECEN LAS LINEAS 390 410

450 470

```

10 REM *****
20 REM * TITULO:REBOTE *
30 REM * VERSION:3.0 *
40 REM * PARA COMMODORE *
50 REM *****
100 REM *****
110 REM *INICIALIZACION DE LAS VARIABLES *
120 REM *****
130 PRINT CHR$(147)
140 LET U=2
150 LET D=18
160 LET L=2
170 LET R=25
180 LET X=4
190 LET M=10
200 LET Y=4
210 LET N=10
220 LET I=1
230 LET K=1
240 LET J=1
250 LET P=-1
260 LET T$=CHR$(19)
270 LET A$=CHR$(17)
280 LET A$=CHR$(29)
300 REM *****
310 REM * RUTINA GENERAL *
320 REM *****
330 GOSUB 630
340 GOSUB 730
350 GOSUB 830
360 LET X=X+I
370 LET M=M+K
380 LET Y=Y+J
390 LET N=N+P
400 GOSUB 900
410 PRINT " ";
420 GOSUB 1000
430 PRINT "0";
440 FOR A=1 TO 50
450 NEXT A
460 GOSUB 900
470 PRINT " ";
480 GOSUB 1000
490 PRINT " ";

```

```

500 GOTO 310
600 REM *****
610 REM * RUTINA DE COMPROBACION DE BORDES 1 *
620 REM *****
630 IF X=R OR X=L THEN LET I=-I
640 IF Y=U OR Y=D THEN LET J=-J
650 RETURN
700 REM *****
710 REM * RUTINA DE COMPROBACION DE BORDES 2 *
720 REM *****
730 IF M=R OR M=L THEN LET K=-K
740 IF N=U OR N=D THEN LET P=-P
750 RETURN
800 REM *****
810 REM * RUTINA DE COMPROBACION DE CHOQUE *
820 REM *****
830 IF X+I<M+K OR Y+J<N+P THEN RETURN
840 LET I=-I
850 LET J=-J
860 LET K=-K
870 LET P=-P
880 RETURN

```

```

10 REM *****
20 REM * TITULO:REBOTE *
30 REM * VERSION:2.0 *
40 REM * PARA IBM-PC Y COMPATIBLES *
50 REM *****
100 REM *****
110 REM *INICIALIZACION DE LAS VARIABLES *
120 REM *****
130 CLS
140 LET U=2
150 LET D=18
160 LET L=2
170 LET R=25
180 LET X=4
190 LET M=10
200 LET Y=4
210 LET N=10
220 LET I=1
230 LET K=1
240 LET J=1
250 LET P=-1
300 REM *****
310 REM * RUTINA GENERAL *
320 REM *****
330 GOSUB 630
340 GOSUB 730
350 GOSUB 830
360 LET X=X+I
370 LET M=M+K
380 LET Y=Y+J
390 LET N=N+P
400 LOCATE Y,X
410 PRINT " ";
420 LOCATE N,M
430 PRINT "0";
440 FOR A=1 TO 50
450 NEXT A
460 LOCATE Y,X
470 PRINT " ";
480 LOCATE N,M
490 PRINT " ";
500 GOTO 310
600 REM *****
610 REM * RUTINA DE COMPROBACION DE BORDES 1 *
620 REM *****
630 IF X=R OR X=L THEN LET I=-I
640 IF Y=U OR Y=D THEN LET J=-J
650 RETURN
700 REM *****
710 REM * RUTINA DE COMPROBACION DE BORDES 2 *
720 REM *****
730 IF M=R OR M=L THEN LET K=-K
740 IF N=U OR N=D THEN LET P=-P
750 RETURN

```

APRENDER CON EL ORDENADOR

```
800 REM *****
810 REM * RUTINA DE COMPROBACION DE CHOQUE *
820 REM *****
830 IF X+I<>M+K OR Y+J<>N+P THEN RETURN
840 LET I=-I
850 LET J=-J
860 LET K=-K
870 LET P=-P
880 RETURN

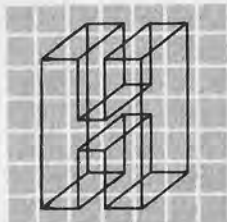
900 REM *****
910 REM * RUTINA PARA SITUACION 1 *
920 REM *****
925 PRINT T$;
930 FOR G=1 TO Y
940 PRINT A$;

950 NEXT G
960 FOR G=1 TO X
970 PRINT D$;
980 NEXT G
990 RETURN

1000 REM *****
1010 REM * RUTINA PARA SITUACION 2 *
1020 REM *****
1025 PRINT T$;
1030 FOR G=1 TO N
1040 PRINT A$;
1050 NEXT G
1060 FOR G=1 TO M
1070 PRINT D$;
1080 NEXT G
1090 RETURN
```

PEQUEÑA HISTORIA DE LA INFORMATICA

■ Otros datos anejos sobre la máquina de Babbage



EMOS hablado bastante sobre máquinas de calcular de todo tipo. Desde máquinas «sencillas» que multiplicaban y dividían sin más pretensiones, hasta máquinas universales, como la Máquina Analítica de Babbage.

También hemos mencionado de pasada el telar de Jacquard, en el que se basó Babbage para resolver algunos problemas fundamentales de su máquina. Efectivamente, las tarjetas perforadas, y el hecho de almacenar al mismo tiempo los datos, y las instrucciones a la máquina constituyen precisamente su modernidad. (Han seguido vigentes hasta hace muy pocos años.)

Babbage basó el sistema de control y «programación» de su máquina en los trabajos de Jacquard, un industrial de Lyon (capital de la seda, en aquel entonces).

Joseph Marie Jacquard nació en Francia, en 1752. Su familia tenía telares, y él heredó su telar tradicional. Nada lo habría hecho pasar a la Historia, pero Jacquard era un investigador, y decidió mejorar su telar, para que fuera capaz de crear los dibujos más complicados y de la forma más precisa. Realmente no fue Jacquard el primero al que se le ocurrió la idea de utilizar tarjetas perforadas para indicar a la máquina la acción que debía llevar a cabo a continuación. La idea ya se le había ocurrido antes a otro francés, Falcon, pero Jacquard la desarrolló y mejoró obteniendo resultados prácticos evidentes.

(Falcon basó su sistema en el que utilizaban algunos antiguos relojes musicales de la

Edad Media. A partir de ese sistema primitivo se han ido creando otros ingenios mecánicos, entre ellos la popular pianola, que toca melodías por un sistema muy semejante al anterior, utilizando fichas perforadas que van unidas y dobladas en forma de acordeón, o mediante rodillos que llevan codificada la melodía, y que pueden sustituirse con facilidad.)

Hasta el momento, cambiar el diseño de las telas o alfombras requería cierto trabajo y tiempo. Mediante las cartulinas, este problema se simplificó enormemente. Las cartulinas llevaban perforaciones codificadas, y utilizándolas el telar realizaba dibujos distintos, lo que hoy llamaríamos «dibujos programables». El telar de Jacquard fue todo un éxito industrial en su época, y venían a Lyon desde toda Europa para verlo funcionar. En la primera década del siglo XIX se habían instalado sólo en Francia más de diez mil unidades.

El funcionamiento de un telar normal hace que los hilos de la urdimbre deban levantarse por cada movimiento de la lanzadera, que es donde se encuentran los hilos de la trama. De esta forma, la lanzadera pasa por debajo de la urdimbre. Para conseguir copiar el dibujo, el sistema prevé que las hebras que se mueven juntas vayan sujetas a la misma varilla. El juego de varillas se va enfrentando cada vez a una tarjeta perforada con el código por cada movimiento de la lanzadera. (A su vez las tarjetas van avanzando en secuencia sobre una cinta, y pasan sobre un tambor para quedar enfrentadas a las varillas.)

Dentro del conjunto de varillas, aquellas que no se encuentren frente a una perforación en la ficha permanecen sin moverse, elevándose las demás. De esta manera, codificando colores, etc., en la tarjeta se pueden conseguir los dibujos más complicados. Babbage tenía en su salón un retrato de Jacquard

PEQUEÑA HISTORIA DE LA INFORMATICA

de un realismo increíble. Para su confección se habían utilizado unas 24.000 fichas perforadas.

Este sistema fue el que copió Babbage para «programar» su máquina. La propia Lady Lovelace describió muy femeninamente que «La máquina analítica tejía dibujos algebraicos del mismo modo que el telar de Jacquard tejía flores y hojas».

Sabemos que Babbage fue un personaje antipático en sus últimos días, y que estaba amargado con la humanidad. Sin embargo, contó con la amistad e incluso con la colaboración de muchas cabezas pensantes de talla de la época. Entre ellas, la primera «programadora» de la historia, Lady Ada Lovelace.

Ada Augusta Lovelace era hija del poeta Byron y de su esposa, Anabella Milbanke. Todos conocemos la belleza de sus versos, y la vida interesante y viajera del poeta. Sin embargo, como marido, dejó mucho que desear. Y, tras haber conquistado a su mujer, la abandonó, junto con su hija Ada. En la familia de su mujer había una gran tradición en el estudio de las Matemáticas, y Anabella no fue una excepción, siendo una aficionada a la materia desde niña. Su marido la llamaba «Princesa de los Paralelogramos». Su hija Ada nació y vivió con la familia de su madre. Su padre jamás se ocupó de ella (probablemente porque pensara que no le sería útil en absoluto, ya que la familia de su mujer era noble, y dada su vida bohemia y atípica, más bien resultaría un estorbo). Lo cierto es que Ada resultó ser una aficionada a las Matemáticas, con una mente clara concreta, y además con ciertas dotes para expresarse por escrito. Desde luego, los textos que nos han quedado de ella son muy superiores a los de Babbage.

Como ya sabemos, Babbage viajó por Europa y publicó algunos trabajos, siempre de lectura difícil y complicada. Entre sus seguidores, un joven militar italiano (que además era ingeniero) dio numerosas charlas sobre los trabajos de Babbage y publicó trabajos, uno de los cuales se imprimió en un diario de Génova en 1842. Los trabajos de Babbage se difundieron muchísimo por Europa gracias a Menabrea.

Lady Lovelace leyó el artículo publicado en el diario de Génova y lo encontró fascinante. Lo tradujo al inglés, añadiéndole numerosos comentarios que aclaraban considerablemente su estudio. Además, tenía un estilo literario fácil, y simplificaba lo que para muchos era complicado. (El propio Babbage ex-

plicaba sus teorías de forma farragosa, y para comprenderlo, resultaba mucho más práctico leer a su amiga Lady Lovelace.)

Ada Lovelace había comprendido perfectamente las ideas de Babbage, y se interesó principalmente por la faceta de hacer llegar a la máquina instrucciones sobre lo que debía hacer. Hoy en día a estas acciones las llamamos «programar», es decir, formular el conjunto de instrucciones que permitan a la máquina realizar su trabajo automáticamente, sin requerir para nada el concurso humano.

Lady Lovelace es muy precisa en todos sus escritos, exponiendo sus puntos de vista con exactitud casi impertinente (punto fundamental para el manejo de la nueva máquina). Parece increíble que esta señora que nació hace más de ciento cincuenta años, desarrollara varios programas para resolver problemas matemáticos, algunos de los cuales eran muy complicados.

Todos los escritos de la dama son interesantes. Sobre la máquina de Babbage escribió lo siguiente: «La máquina excede a las predecesoras en la amplitud de los cálculos que puede ejecutar, la facilidad, seguridad y precisión, así como en la ausencia de la necesidad de que intervenga la inteligencia humana durante la ejecución de los cálculos».

Lady Lovelace llegó a conocer la máquina casi mejor que su creador. Había estudiado su arquitectura hasta el límite, lo que le permitió aconsejar a Babbage algunas modificaciones, que éste aceptó como posibles.

Sin embargo, la frase que mejor demuestra su conocimiento profundo de la máquina de Babbage y su filosofía es la siguiente: «La máquina Analítica no tiene pretensiones de originar nada. Sólo puede hacer aquello que nosotros sepamos ordenarle que ejecute. Puede seguir el análisis pero no tiene poder de anticipar relaciones analíticas o verdaderas cualesquiera. Su dominio es ayudarnos para hacer disponible lo que ya conocemos». ¿Existe otro modo más claro de explicar una máquina tan compleja?

En 1979, propiciado por el Departamento de Defensa de los Estados Unidos, se trabajó en un nuevo lenguaje en Francia. (Su principal creador fue Jean Ichbiah.) Recibió el nombre de Ada, en reconocimiento a la labor de estudio y difusión de esta dama inglesa.

En el siglo XIX se continuó intentando mejorar las máquinas de calcular, abandonando casi totalmente la idea de máquina univer-

sal de Babbage. Se abandonó el antiguo sistema de palancas y giros para pasar a utilizar teclas, que eran mucho más cómodas y rápidas. La primera máquina de cálculo que utilizaba teclas apareció en el mercado hacia 1850, y era americana. Su inventor fue D. D. Parmelee. Sin embargo, era una máquina muy incómoda, que sólo sumaba una columna de números cada vez. Además la respuesta de las teclas no estaba muy controlada, y en ocasiones, al presionar inadecuadamente, se podían obtener respuestas inexactas. (Esta calculadora era del tipo de «cremallera», naturalmente con sus correspondientes piñones y engranajes de transmisión.)

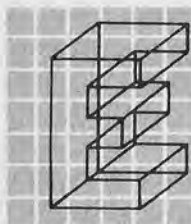
Algo más tarde, apareció en el mercado otra máquina que podía sumar varias columnas a la vez. Sin embargo, tampoco fue una máquina segura, a pesar de los esfuerzos de mejorarla de su «inventor», Thomas Hill.

La máquina más segura y eficaz (real-

mente fiable) no vio la luz hasta 1874. Su inventor fue un sueco, Thomas Ohdner. Esta máquina tuvo un éxito enorme, por toda Europa, incluyendo los aislados países centroeuropeos y Rusia. Otro prototipo muy similar apareció en los Estados Unidos patentado por Baldwin y Monroe. También tuvo un gran éxito, y fue la base para la creación de una Compañía de Calculadoras Monroe, que existe en nuestros días.

En 1885 aparecería otra máquina a teclas pero mucho más fiable que los primeros intentos llevados a cabo hasta el momento. Su creador, Felt, la llamó Contómetro. También tuvo éxito por su fiabilidad, y por el apoyo que la empresa que creó daba al cliente, creando manuales, dando clases prácticas de manejo, etcétera. Su máquina llegaba a hacer raíces cuadradas, calculaba porcentajes rápidamente, etcétera.

■ Proceso paralelo



Es muy frecuente en aplicaciones en tiempo real que los trabajos a realizar requieran cálculos masivos. De esta característica ha derivado el considerable auge que va tomando el proceso en paralelo. Existen innume-

rables campos para los que su desarrollo tiene enorme interés (recordemos, por ejemplo, los trabajos sobre Visión Artificial, Astronomía, Predicción Meteorológica, etc.). Se han hecho estudios para mejorar el hardware de las máquinas (la arquitectura concretamente, pero los progresos en el campo del proceso en paralelo han sido bastante limitados).

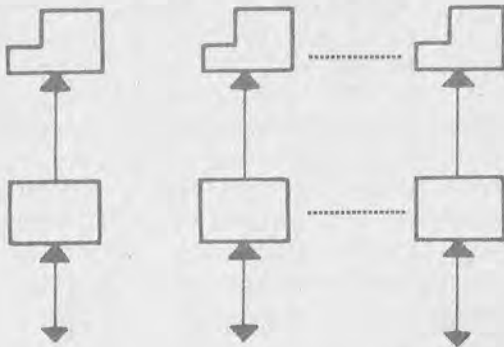
Hablaremos, pues, sobre el software desarrollado para estos sistemas, y también de la metodología y sistemas automáticos asociados, que facilitan las mejoras del proceso en paralelo cambiando las particiones de dicho paralelismo, y analizando después los resultados.

Al principio, la programación en paralelo estaba motivada por la necesidad de reducir los tiempos de ejecución, que en determinados procesos eran muy largos, y que cuando se trabajaba en tiempo real era perentorio reducirlos. El tiempo de ejecución es largo principalmente debido al enorme cúmulo de datos a procesar, y no a la complejidad de las operaciones a realizar con ellos. Así pues, el paralelismo en las arquitecturas de los ordenadores de gran tamaño se ha concebido como un sistema de mejorar la capacidad de proceso a nivel de las operaciones primitivas. Se han desarrollado, por ejemplo, compiladores especiales adaptados para que acepten como entrada programas convencionales (es-

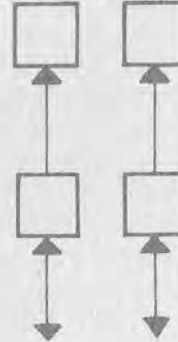
critos fundamentalmente en Fortran) y reconozcan las operaciones que pueden ser realizadas en paralelo (para una arquitectura de ordenador específica). Además, en las máquinas de flujo de datos, las operaciones primitivas individuales se organizan en el tiempo en los respectivos procesadores individuales. La partición en módulos muy pequeños (granularidad muy fina) de estos programas, lleva a sobrecargas (overhead) en comunicaciones, y cuellos de botella, limitando efectivamente el alcance de este posible paralelismo.

La situación es la siguiente: en un extremo, se puede hacer que la granulación sea muy fina (módulos muy pequeños) y las operaciones se asignan a los procesadores respectivos. Con ello, efectivamente se maximiza el paralelismo, pero el resultado es una sobrecarga del sistema (overhead) en comunicaciones y en el acceso a la memoria. En el otro extremo, si la granulación es más basta (los módulos de los programas mayores), el proceso sólo se parte en algunos gránulos, el paralelismo es menor, pero la sobrecarga deja de ser preocupante. Otra posibilidad es mejorar lo anterior, programando a varios niveles, e incrementando la granulación, para ir obteniendo elementos más finos cada vez. Para nuestro trabajo vamos a suponer que estamos trabajando con una red de comunicaciones compuesta por varios procesadores, cada uno de los cuales dispone de su propia memoria, independientemente de que cada procesador (en cada nivel) sea internamente un procesador unitario secuencial, o esté formado por procesadores paralelos. Así pues, supondremos que cada gránulo está asignado a un único procesador. Nuestra idea consiste en intentar obtener en cada nivel el equilibrio entre el paralelismo y el overhead (sobrecarga) en comunicaciones. Para acercarnos a nuestro ob-

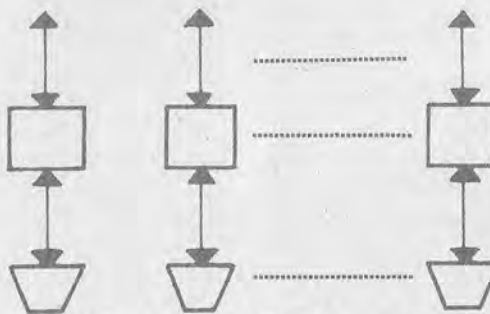
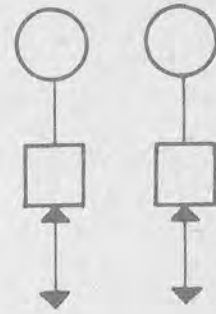
**Interfaces con los
terminales de los Operadores**



**Datos de
Modelización e
Históricos**



**Datos de
Entrada-Salida
y de planificación**



Sensores y Actuadores (Interfaces y Equipos)

Configuración general típica de un sistema multiproceso donde aparecen los distintos módulos y sus interconexiones.

jetivo será necesario mantener la adecuada correspondencia entre la progresiva partición del problema (que realicemos a nivel de programación) y la distribución en gránulos de programas paralelos. Esta correspondencia es necesaria para el desarrollo adecuado de las pequeñas partes, así como para la depuración de los programas, modificaciones y extensiones del software total, a lo largo de todo su ciclo de vida.

Las particiones del programa se basan, inicialmente en la afinidad funcional. Se emplearán series de sistemas automáticos para proporcionar eventualmente realimentación acerca de las capacidades disponibles. Esto, junto a las limitaciones en cuanto al tiempo, derivadas del entorno operacional, irán guiando al programador a realizar otras particiones, o a ir consolidando los módulos, hasta llegar a la mejor división, aquella cuyos resultados sean satisfactorios. Estas etapas son muy parecidas a las empleadas en desarrollar sistemas en tiempo real, pero, eso sí, en nuestro caso, el sistema que se está programando es muy complejo, los datos a tratar son tantos que el programador se perdería si no siguiera para resolver el problema ciertas etapas automáticamente.

a) Después de la partición inicial del problema, se configura una red de módulos de comunicaciones, creando un mapa de comunicaciones entre los procesadores paralelos. Ya hemos dicho antes que cada procesador puede ser a su vez un subsistema multiproceso.

b) A continuación, se pasa a ir definiendo cada uno de los gránulos independientemente, compilándolos en los programas respectivos, y probándolos.

c) Inmediatamente, se pasa a analizar los retardos (delays) dentro de cada gránulo entre la entrada y/o la salida. Estos retardos se deben a una combinación de operaciones y comunicaciones con los otros módulos.

d) Los módulos se integran dentro de la configuración específica. Se evalúa en ese momento la capacidad total de la red, considerando las limitaciones en tiempos.

e) Se sigue considerando la configuración, consolidando los respectivos módulos o dividiéndolos, para tratar de eliminar el cuello de botella o los retardos que no pueden tolerarse. Durante esta revisión, puede que necesite la adición de más procesadores a la configuración, si es que se pasa a dividir alguno de los elementos. Sin embargo, también, puede ocurrir que se integren varios elemen-

tos dentro de un único módulo de proceso, o dentro del mismo procesador.

f) Se repiten los pasos hasta que la configuración sea satisfactoria. En esta etapa es necesaria la colaboración humana para aportar otras consideraciones externas al problema, como caminos críticos que signifiquen unas restricciones de tiempos. Además, las redes locales suelen tener normalmente caminos circulares, que requieren el análisis de su comportamiento, por parte de un programador.

Para llevar a cabo las etapas anteriores, se necesitan tres tipos de sistemas automáticos:

a) Un Configurador, para sintetizar los módulos componentes en una red que realice los cálculos deseados. La configuración no debe ser tan rígida que apenas pueda modificarse, por el contrario, debe ser maleable para facilitar los cambios durante el ciclo de cómputo programado. También debe ser ejecutable progresivamente, para sus posibles comprobaciones (quizá en una máquina especial de desarrollo), y también, en ciertos casos en el propio superordenador paralelo en el que al final será ejecutado.

b) Un lenguaje para definir cada módulo, y un compilador para traducirlo en programas que irán a los respectivos procesadores. Respecto de la eliminación de errores, es útil que se realice en cada módulo individualmente. El lenguaje debe soportar también cierta modularidad interna, por posibles particiones anteriores a la consolidación de cada módulo.

c) También será necesario un TIMER, para procesar los retardos propios de los gránulos.

Veamos alguna aplicación concreta.

Hasta el momento, los grandes ordenadores se han utilizado principalmente para cálculos masivos autónomos, tales como simular fenómenos físicos o sociales. Sin embargo, en este caso vamos a tratar fenómenos que se desarrollan e interactúan en tiempo real. Verdaderamente, estos últimos problemas son los que han llevado directamente al uso del paralelismo en computación. El problema más importante radica en la criticidad de los tiempos, mucho más que en el cúmulo de datos.

En el dibujo, ilustramos de forma muy general este tipo de problemas. Los rectángulos denotan módulos de proceso (gránulos) a nivel basto. Como puede verse, pertenecen a cuatro clases diferentes. A la izquierda se en-

cuentran los módulos que sirven para comunicar a los operadores humanos con el conjunto del sistema. Estos gránulos solicitan comandos o preguntas de los terminales de operación, y los dirigen a los módulos destino respectivos. También reciben información para los correspondientes terminales. Los interfaces con los sensores (que aparecen en la parte inferior de la figura) obtienen y compactan la información obtenida, enviándola a otros gránulos. También interpretan comandos de alto nivel de otros módulos y los traducen en una secuencia de pasos para el equipo del actuador. En el centro se pueden ver gránulos que modelan el entorno externo, y proyectan los posibles efectos de los distintos planes alternativos. Estos módulos también mantienen datos antiguos. A la derecha se encuentran módulos que planean y controlan la actividad de los actuadores. Estos módulos optimizan el uso de recursos, procesan tareas especificadas en la entrada y producen informes de salida. A pesar de que todos los gránulos son capaces de comunicarse unos con otros directamente, debe establecerse una relación jerárquica para las comunicaciones entre ellos, y puede ser útil limitar éstas para que sólo se lleven a cabo entre los elementos más próximos.

La figura que mostramos puede ser el esquema, por ejemplo, de una factoría integrada y automatizada. Los operadores inician y completan las actividades, observando las distintas etapas, y cómo va progresando la fabricación. Los actuadores se ocupan del manejo del material y de las operaciones de producción. Por otro lado, los sensores proporcionan imágenes del proceso, fuerza y, en general, la información necesaria para coordinar todo el proceso de fabricación. Los módulos de simulación mantienen el estado general del equipo, e informan sobre posibles alternativas a tomar, para optimizar el uso de los recursos, evitando errores importantes. Finalmente, los módulos de planificación reciben los objetivos generales de fabricación y planean su ejecución.

La figura anterior, además de poder ser el esquema de una fábrica, también puede ser el esquema del sistema de control de un aeropuerto. Los operadores pueden introducir o modificar planes de vuelo, ver imágenes de radar y recibir informes sobre el estado general del sistema o algunas de sus partes. Los sensores, por su parte, pueden suministrar información sobre el estado del tiempo, veloci-

dad, altura, etc. Los actuadores serán los que controlen el vuelo. De esta forma, los gránulos de simulación mantendrán y proyectarán la ruta, mientras que los gránulos de planificación evaluarán otras posibles alternativas.

Existen también tareas que, aunque no suponen restricciones respecto al tiempo (no son en tiempo real) sí suponen interacciones con personas relacionadas con esos estudios. A pesar de todo ello, también puede identificarlas la figura anterior, si se eliminan los módulos de interfaz con los actuadores.

Los módulos individuales se pueden considerar como elementos asíncronos, que funcionan con la llegada de la información. Esta información puede guardar cola a su llegada a los elementos individuales. Los retardos se pueden determinar basándonos en las relaciones de envío y recepción de la información. Los elementos de modelación y planeamiento consumen la información producida por el sensor y el actuador, sirviendo de interface con los módulos y produciendo respuestas. El tiempo en que se producen las respuestas es crítico para que el uso del equipo sea eficaz, y para evitar errores, y problemas de todo tipo. Los cálculos masivos serán necesarios en ciertos módulos que actúan de interfaz con algún sensor, como, por ejemplo, en el análisis de imágenes y en la modelización de planes. Pueden suponer retrasos enormes, que se podrán reducir creando nuevas particiones de gránulos para incrementar el paralelismo o, en algunos casos, agrupando éstos para reducir las comunicaciones. Los retardos permisibles vienen dictados por el entorno externo, y por las características del equipo, no especificadas en el cálculo. Para evaluar todos estos problemas es necesario contar con la co-

laboración de especialistas, diseñadores, que puedan valorar adecuadamente todos esos parámetros impuestos.

En la última década se han escrito numerosos lenguajes para facilitar el diseño, implementación, especificación y verificación de programas paralelos. Los lenguajes de configuración suponen un medio sencillo y eficaz para sintetizar un conjunto de gránulos para formar un programa en paralelo. Las especificaciones de configuración, escritas en un lenguaje de configuración, suponen una expresión centralizada del programa paralelo, importante en el momento de presentar y comprender la abstracción que debe soportar el programa. El aspecto clave de un configurador es separar aquellos aspectos que han de ser resueltos mediante programación paralela de aquellos otros que se pueden resolver mediante programación secuencial. Esta separación, hace que un lenguaje de configuración sea independiente de los lenguajes de programación secuenciales, y así permite a los módulos componentes estar escritos en distintos lenguajes de programación, adecuados a sus distintas aplicaciones.

Los configuradores sirven para proporcionar toda la información necesaria para construir y ejecutar un programa paralelo. Por ejemplo, identifican los tipos de módulos de software con los que se construirá el programa, los formatos de estos módulos, y cómo esos módulos están interconectados. También los recursos de hardware necesarios para la ejecución del programa, dónde deben cargarse los gránulos, los requerimientos de tiempo que debe satisfacer el programa, y cómo manipularán la información los distintos gránulos que se encuentran dentro del mismo procesador.

Actuador. Equipo periférico de acción capaz de realizar funciones en el área de la robótica, de acuerdo con las instrucciones recibidas por el programa.

Caminos (paths). En comunicaciones, canal que une varios puntos en la red: este canal puede estar formado por varias etapas (y hasta puede incluir ordenadores u otros dispositivos).

Cola. Una cola es simplemente un espacio reservado en la memoria (generalmente la memoria intermedia) en el que se retiene una lista de información temporalmente.

Lista FIFO (primero en entrar, primero en salir). Pila de desplazamiento ascendente. En uno de los extremos se hacen todas las inserciones, y en el otro todas las extracciones. Esta cola se puede implementar en el soporte físico como un tipo de memoria especial, carente de direcciones. Se utiliza normalmente para acelerar la introducción/salida de la memoria intermedia de un flujo de datos en tiempo real y es una forma de memoria que necesita tiempo de comienzo/parada. Las colas se caracterizan por la forma en que se van uniéndose a ella los procesos esperando la utilización del servicio de que se trate así como también por cómo se efectúa la selección de aquel elemento de ésta que va a pasar a procesarse. Esta decisión depende del elemento Administrador de la Cola.

Configuración, lenguaje de. Lenguajes específicos concebidos para el diseño de soluciones en ordenadores de proceso en paralelo.

Delay. (Véase **Retardo.**)

Granularidad. Tamaño de los módulos en que se divide un programa para su posterior proceso en paralelo.

Overflow. Desbordamiento. Situación que se produce cuando una operación excede la capacidad de almacenamiento de la zona de memoria preparada para almacenarlo. También puede referirse únicamente a la parte de la información que excede la capacidad de almacenamiento de la memoria reservada para ese fin.

Overhead. (Véase **Sobrecarga.**)

Overlap. Solapamiento. Cierta forma de paralelismo en el que los sucesos tienen lugar concurrentemente. Con el fin de incrementar el rendimiento del ordenador se van solapando las instrucciones, y, por ejemplo, se va buscando una, mientras se está procesando la anterior, etc. Si este solapamiento se produce en cálculos aritméticos elementales, el proceso recibe el nombre de *pipelining* (conducción, tubería).

Existe un tipo de memorias (llamadas solapadas) que permite recuperar palabras sucesivas desde bancos distintos. De este modo se solapan las operaciones lógicas (que se realizan en un banco), la reescritura (en otro), y el acceso a la memoria (en otro distinto también).

Proceso en paralelo. Este término admite bastantes conceptos, todos ellos relacionados. Proceso en el que en un momento dado, se está ejecutando más de un proceso, o cuando en un momento dado están activos varios procesadores para realizar un conjunto de procesos, simultáneamente. Es frecuente, sin embargo, llamar proceso en pa-

ralelo a una situación en la que potencialmente están activos varios procesos aunque físicamente sólo se esté ejecutando uno en cada momento.

Proceso Pipeline. Proceso por etapas. Procedimiento para acelerar la ejecución de los procesos. En un programa cualquiera, cada instrucción se divide en muchas etapas cortas y simples, de forma que en un momento dado, se tienen muchas instrucciones en varias etapas. De esta forma, cuando la primera etapa ha terminado, los resultados pasan a una segunda, que emplea otro soporte físico diferente, quedando el primer soporte físico libre para seguir el proceso. Con este sistema, se aumenta el rendimiento enormemente en los procesos secuenciales, complicando por el contrario la unidad de control que lleva la cuenta de todas las cantidades que se están realizando en un momento dado. Esta técnica se utiliza en grandes ordenadores, que manejan gran-

des cantidades de datos en los que el tiempo empleado es factor fundamental. Otra forma de enfocar el problema, es el proceso en paralelo.

Programación en paralelo. Creación de un programa que contiene varios módulos que han de ejecutarse en paralelo (al mismo tiempo). En muchas ocasiones, pasos de programa que deberían realizarse de forma secuencial se realizan utilizando varios procesadores.

Sobrecarga (Overhead). Tiempos adicionales que emplea un sistema en acciones tales como asignar recursos, dar respuesta a estados especiales, protecciones, acciones para asegurar la fiabilidad, etc. Estas acciones no contribuyen directamente al progreso de las tareas del usuario del sistema.

Transmisión en paralelo. Transmisión (normalmente simultánea) de los caracteres de una palabra sobre diferentes líneas.

Instrucciones, registro de. Registro que almacena una instrucción para su ejecución.

Instrucciones, repertorio. Conjunto de instrucciones de que dispone un procesador.

Conjunto de operaciones que se pueden representar por un código de operación dado.

Integrado, proceso de datos. Proceso de datos en el que la coordinación de la adquisición de los datos, y otras etapas de su proceso se realizan por un sistema coherente. Proceso de datos para la empresa u otra organización, en el que los datos se integran en un sistema único de tal modo que las distintas actividades se interconexionan para transmitirse los datos y dar resultados unitarios.

Integrador, motor. Motor diseñado para dar una relación constante entre la velocidad de rotación del eje (salida) y la señal de entrada. De este modo, el ángulo de rotación del eje con respecto a una posición dada de origen es proporcional a la integral de tiempo de la señal aplicada.

Integrador. Dispositivo cuya función de salida es proporcional a la integral de la función de entrada con respecto a una variable especificada, como puede ser un medidor del número de vatios consumidos por hora.

Inteligencia artificial. Capacidad que dispone un dispositivo por la cual es capaz de realizar funciones asociadas normalmente con la inteligencia humana. Entre ellas, podemos considerar el razonamiento, el aprendizaje y la capacidad de superación.

Interblock gap. Significa lo mismo que el espacio entre bloques (block gap).

Interface. Término inglés que se refiere a alguna frontera común entre dos dispositivos o elementos. Puede referirse a un componente de hardware que sirve de unión entre dos dispositivos, o puede ser una parte de los registros de almacenamiento, accesible por dos o más programas de ordenador. Muy normalmente se utilizan interfaces de software que realizan la misma tarea indicada para los interfaces hardware.

Interleave. Término inglés que indica la acción por la que se ordena una secuencia de elementos o acontecimientos de forma que se intercalen con una o más secuencias del mismo tipo, de forma que cada secuencia mantenga su identidad. Por ejemplo, para organizar un almacenamiento entre bancos de datos, de forma que las referencias secuenciales de datos se puedan sobreponer en un período dado de tiempo.

Interno, almacenamiento. Almacenamiento direccionable, controlado directamente por la unidad central de proceso de un ordenador digital.

Interpelativo código. Repertorio de instrucción, para la entrada de lenguaje fuente a un intérprete.

Intérprete. Programa de ordenador que traduce y ejecuta cada instrucción de lenguaje fuente antes de traducir y ejecutar la siguiente instrucción.

En particular, en los microprocesadores domésticos suele existir nativo un intérprete de BASIC para traducir instrucciones de BASIC a lenguaje máquina. Tiene una función análoga, a la de los compiladores lenguajes más complejos, como COBOL o FORTRAN. El programa intérprete forma parte

del software de base de un ordenador y siempre debe encontrarse en la memoria, para poder «interpretar» los programas en el momento de su ejecución. En algunos ordenadores, el intérprete está cargado de forma fija en la memoria ROM, pero en otros, por el contrario, se carga cada una de las veces en la memoria central, RAM, desde algún soporte exterior (disquetes o cintas magnéticas, etc).

Inter-record gap. Término inglés que denota el espacio de separación entre dos registros.

Interrupción. Detención de un proceso de forma que pueda reiniciarse en otro momento. También puede referirse a una señal de hardware que interrumpe la ejecución normal de un programa, permitiendo la ejecución de otros programas o subrutinas. Al término de estas ejecuciones, se reanuda el proceso anterior.

Interrupción, máscara de. Sistema para suprimir las interrupciones de forma selectiva cuando se producen, de tal forma que se pueda actuar sobre ellas en un momento posterior.

Interrupciones, prioridades en las. Orden de importancia asignado a las interrupciones. Un sistema puede responder únicamente a una interrupción cada vez, o puede tener una tasa más alta que la de servicio. El sistema puede disponer de máscaras de interrupción, que supriman algunas selectivamente, si ha ocurrido alguna interrupción cuyo orden de prioridad sea mayor.

Dispositivo que interrumpe una determinada señal o acción. Existen dos tipos de interrupciones en informática, las de hardware (interruptores físicos) y las de software, que pueden utilizar simplemente una variable binaria.

Intervalo. Espacio entre dos elementos. Zona de la cinta magnética entre dos grabaciones o registros sucesivos, que no contiene información alguna, y que actúa como separador. Zona de un disco rígido o flexible utilizada como separador entre dos informaciones, y no accesible por el usuario.

Introducción de los datos. Información proporcionada desde el exterior a un dispositivo de proceso de datos.

Invariante. Propiedad que permanece verdadera sean cuales fueren las transformaciones llevadas a cabo. Se puede aplicar a programas, a módulos o a operaciones individuales.

Inversor. Parte lógica electrónica que invierte las señales que recibe de forma que cuando recibe un 1 lógico (verdadero) lo convierte en un 0 lógico (falso), y viceversa. Implementa, por tanto la operación lógica NO.

Invertido, fichero. En la recuperación de la información, método de organización de ficheros, con índices cruzados, en el cual una determinada clave identifica un registro, y los elementos, números o documentos relacionados con esa palabra clave. También puede referirse a un fichero cuya secuencia ha sido invertida.

ISDN. Abreviatura de Integrated Services Digital Network (Red Digital de Servicios Integrados). Red Internacional de comunicaciones digital. Es portadora de imagen, sonido, datos informáticos, etc. Está concebida como un sistema único capaz de dirigir las comunicaciones de todo tipo.

ISO. Siglas de International Standard Organization (Organización Internacional de Normas). Organismo que marca las normas internacionales que deben cumplir todos aquellos dispositivos y sistemas, etc., relacionados con la Informática. Regula desde los tamaños de tornillos hasta protocolos de enlaces, normas de codificación, etc. Fue fundada en 1946 por un comité en el que participaron más de 60 países.

Item. Término inglés para designar un elemento, como miembro de un grupo, etc. Así pues, cada elemento forma parte del grupo correspondiente. También puede referirse a una colección de caracteres relacionados, considerados como una unidad.

Iteración. Repetición de un grupo de instrucciones o de un programa, hasta que se verifique que se da una determinada condición de control. (También se llama Bucle).

I/O. Siglas que se corresponden con las españolas E/S/ (Entrada/Salida). Pueden aplicarse a multitud de términos informáticos (canal de E/S, control E/S, dispositivo E/S, bus E/S, puerta E/S, etc).

